ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Electrónica Industrial y Automática

INTEROPERABILIDAD DE MODELOS HARDWARE: UN ADAPTADOR DE OSLC PARA QUARTUS II

Sara Martín Fraile

Tutor: José María Álvarez Rodríguez

AGRADECIMIENTOS

Este trabajo, ante todo, va dedicado a mi madre y a mi padre. Valoro muchísimo todos los esfuerzos que habéis tenido que hacer para que lo consiga, gracias por ser como sois, sin vuestro apoyo no hubiera llegado hasta aquí.

A Alex, gracias por compartir tantos momentos conmigo, tanto buenos como malos, fuimos un equipo desde el principio y todo lo que hemos conseguido ha sido juntos, tirando el uno del otro. Gracias por tu apoyo.

"Casi todos piensan que una mente brillante es la principal cualidad de un buen científico. No están en lo cierto: es su actitud."
Albert Einstein

Tabla de contenido

ÍNDICE DE ILUSTRACIONES **9**

ÍNDICE DE TABLAS **10**

ACRÓNIMOS Y DEFINICIONES **11**

RESUMEN **13**

ABSTRACT **14**

1. INTRODUCCIÓN **15**

1.1. OSLC **15**

1.2. Quartus II **18**

1.3. Motivación y objetivos **19**

2. ESTRUCTURA DEL DOCUMENTO **21**

3. ESTADO DEL ARTE **22**

3.1. Industria 4.0 ..... **22**

3.2. Comunicación y OSLC **23**

3.3. Interoperabilidad **26**

3.3.1. Como alcanzar la interoperabilidad 27

3.3.2. Problemas con la interoperabilidad 28

3.4. Integración **29**

4. ANÁLISIS DEL SISTEMA **31**

4.1. Planteamiento del problema **31**

4.2. Requisitos **32**

4.3. Casos de uso **35**

5. DISEÑO DEL SISTEMA **36**

5.1. Diseño abstracto **36**

5.2. Tecnología y entorno de desarrollo **37**

5.2.1. .NET 37

5.2.2. API CAKE 38

5.2.3. Knowledge Manager 39

5.3. Transformación de VHDL a SRL **39**

5.3.1. Metamodelo VHDL 39

5.3.2. Metamodelo SRL 44

5.3.3. Reglas de transformación 47

6. IMPLEMENTACION Y PRUEBAS **49**

6.1. Componentes del sistema **49**

6.2. Diagrama de bloques y VHDL **49**

6.3. API CAKE **51**

6.4. Knowledge manager **52**

6.5. Pruebas **53**

7. MARCO LEGISLATIVO Y SOCIOECONÓMICO **55**

7.1. Marco regulador y aspectos legales **55**

7.2. Entorno socioeconómico **57**

8. PLANIFICACIÓN 58

9. PRESUPUESTO 61

9.1. Coste personal 61

9.2. Coste de material 62

9.2.1. Costes de Hardware 62

9.2.2. Costes de Software 63

9.3. Costes indirectos 64

9.4. Coste total del proyecto 64

10. CONCLUSIONES Y TRABAJO FUTURO 65

10.1. Motivación 66

11. REFERENCIAS Y BIBLIOGRAFÍA 67

11.1. Referencias 67

11.2. Bibliografía 69

ÍNDICE DE ILUSTRACIONES

Ilustración 1: Interoperabilidad OSLC 15
Ilustración 2: Interoperabilidad entre herramientas 16
Ilustración 3: Etapas del proceso de diseño en la plataforma ALTERA 18
Ilustración 4: Integración 19
Ilustración 5: Tipos de comunicación 24
Ilustración 6: Interoperabilidad 26
Ilustración 7: Ejemplo tipo aplicación OSLC 32
Ilustración 8: Casos de uso 35
Ilustración 9: Diseño abstracto del sistema 36
Ilustración 10: Tecnología de la herramienta 37
Ilustración 11: Logo Microsoft .NET 37
Ilustración 12: Knowledge Manager 39
lustración 13: Logo VHDL 39
Ilustración 14: VHDL - introduction 40
Ilustración 15: Jerarquía VHDL 41
Ilustración 16: Ejem. 1 Puerta lógicas con VHDL (Entidad) 42
Ilustración 17: Ejm. 2 Puertas lógicas con VHDL (Arquitectura) 42
Ilustración 18: Sentencias concurrentes VHDL 43
Ilustración 19: Como se escriben las sentencias concurrentes 43
Ilustración 20: Sentencia IF - Sentencia secuencial 44
Ilustración 21: Componentes del sistema 49
Ilustración 22: Captura de pantalla de la impresión del análisis de VHDL 50
Ilustración 23: Diagrama de Gantt 59

ÍNDICE DE TABLAS

Tabla 1: Ejemplo tabla requisitos 32
Tabla 2: Requisito funcional R-01 33
Tabla 3: Requisito funcional R-02 33
Tabla 4: Requisito funcional R-03 33
Tabla 5: Requisito funcional R-04 33
Tabla 6: Requisito funcional R-05 33
Tabla 7: Requisito funcional R-06 34
Tabla 8: Requisito funcional R-07 34
Tabla 9: Requisito funcional R-08 34
Tabla 10: Requisito funcional R-09 34
Tabla 11: Metadatos SRL 46
Tabla 12: Transformación VHDL a SRL 48
Tabla 13: Tabla ejemplo pruebas 53
Tabla 14: Prueba PR-01 54
Tabla 15: Prueba PR-02 54
Tabla 16: Prueba PR-03 54
Tabla 17: Prueba PR-04 54
Tabla 18: Total de días y horas trabajadas 61
Tabla 19: Coste personal 62
Tabla 20: Coste material 63
Tabla 21: Costes indirectos 64
Tabla 22: Coste total del proyecto 64

ACRÓNIMOS Y DEFINICIONES

- **ALM:** Application Lifecycle Management.
- **API:** Application Program Interfaces.
- **ASIC:** Application-specific integrated circuit.
- **CC:** Creative Commons.
- **CPLD:** Complex Programmable Logic Device.
- **CPS:** Creative Process Solving.
- **CRUD:** Create, Read, Update and Delete.
- **ECMA:** European Computer Manufacturers Association.
- **Estándar abierto:** es una especificación técnica (acrónimos) (es decir un conjunto de requerimientos técnicos de funcionalidad)
- **Framework:** Conjunto de herramientas y API's para desarrollar software.
- **HDL:** Hardware Description Language.
- **HTML:** HyperText Markup Language.
- **HTTP:** Hypertext Transfer Protocol.
- **IDE:** Integrated Development Environment.
- **IEEE:** Instituto de Ingeniería Eléctrica y Electrónica.
- **ISO:** Open System Interconnection.
- **KM:** Knowledge Manager.
- **LPI:** Ley de Propiedad Intelectual.
- **OSLC:** Open Services for Lifecycle Collaboration.
- **PLM:** Product Lifecycle Management.
- **RAND:** Reasonable and non- discriminatory.
- **RDF:** Resource Description Framework.

- **REST:** Representational State Transfer.
- **RTL:** Register Transfer Level.
- **Smart grid:** Red eléctrica inteligente (o REI; smart grid en inglés).
- **SRL:** System Representation Language.
- **URI:** Uniform resource identifier.
- **Vendor lock-in:** El uso restringido o propietario de una tecnología, solución o servicio desarrollado por un proveedor.
- **Verilog:** Es un lenguaje de descripción de hardware.
- **VHDL:** Lenguaje de especificación.
- **VHSIC:** Very High Speed Integrated Circuit.
- **W3C:** World Wide Web Consortium.

RESUMEN

El presente documento forma parte del Trabajo de Fin de Grado: Interoperabilidad de modelos hardware: Un adaptador de OSLC para Quartus II.

Con este proyecto se pretende formar parte de la comunidad de OSLC. Esta comunidad esta formada por un grupo de desarrolladores y organizaciones de software que trabajan para poder estandarizar la forma en que las herramientas del ciclo de vida del software comparten datos.

Open Services for Lifecycle Collaboration (OSLC) es un conjunto de especificaciones pensadas para simplificar la integración de herramientas en el ciclo de vida de distribución de software. Se ha desarrollado en respuesta a viejos obstáculos para una integración efectiva de productos de ciclo de vida; OSLC permite crear integraciones a gran escala y de fácil mantenimiento en un entorno de herramientas heterogéneo. **[1]**

Lo que queremos es poder unificar herramientas para la optimización del trabajo. Para ello vamos a crear una herramienta que funcione como adaptador de OSLC para Quartus II. El objetivo principal es ser capaces de representar y reutilizar modelos físicos en VHDL. Esto lo haremos transformando a SRL y utilizando las APIs necesarias.

El trabajo está formado por dos componentes diferenciados, en primer lugar la aplicación desarrollada para conseguir el adaptador de OSLC para Quartus II, y en segundo lugar, este documento, en el que se explicará la parte técnica y desarrollo de dicha aplicación.

ABSTRACT

This document is part of the Final Degree Project: Interoperability of hardware models: An OSLC adapter for Quartus II.

This project aims to be part of the OSLC community. This community is made up of a group of developers and software organizations that work to standardize the way in which software lifecycle tools share data.

Open Services for Lifecycle Collaboration (OSLC) is a set of specifications designed to simplify the integration of tools in the software distribution life cycle. It has been developed in response to old obstacles for effective integration of life cycle products; OSLC allows large-scale and easy-maintenance integrations to be created in a heterogeneous tool environment. **[1]**

What we want is to be able to unify tools for work optimization. To do this we will create a tool that works as an OSLC adapter for Quartus II. The main objective is to be able to represent and reuse physical models in VHDL. We will do this by transforming SRL and using the APIs needed.

The work consists of two differentiated components, first of all the application developed to obtain the OSLC adapter for Quartus II, and secondly, this document, in which the technical part and development of said application will be explained.

1. INTRODUCCIÓN

1.1.OSLC

En 2008 se originó Open Services for Lifecycle Collaboration (OSLC), una comunidad abierta que buscan la integración del desarrollo de software mediante cúmulo de especificaciones definidas.

Es una comunidad que esta en constante evolución como por ejemplo a áreas como ALM y PLM.

La intención es facilitar la vida de los usuarios de herramientas y proveedores de herramientas, facilitando que las herramientas trabajen juntas. **[1]**

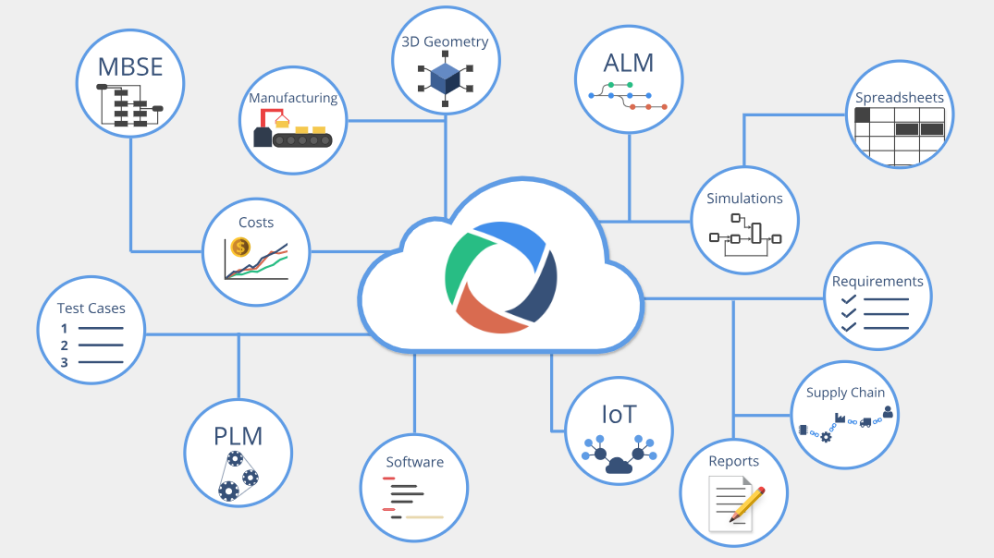


Ilustración 1: Interoperabilidad OSLC

OSLC está abierto en el sentido de que cualquiera puede participar (por ejemplo, en grupos de usuarios). En apoyo de la iniciativa OSLC, existen proyectos de código abierto para construir una implementación de referencia OSLC y conjuntos de pruebas para varios lenguajes de programación y marcos.

Para crear integraciones independientes de las APIs del producto, hay un método estándar proporcionado por las especificaciones de OSLC mediante datos enlazados y protocolos WWW.

Las incompatibilidades de herramientas y el verse encerrado en ciertos productos o versiones específicas son problemas que las especificaciones de OSLC quieren reducir, eliminando la necesidad de conexión entre los tipos de datos en aplicaciones y buscando la integración.

OSLC permite acceder a los datos en tiempo real y elimina las barreras entre herramientas. Con OSLC también es mucho más sencillo el acceso a los datos, ya que otras integraciones requieres complejas instalaciones.

Las herramientas que utilizan especificaciones OSLC pueden mantener fácilmente integraciones de herramientas de diversos proveedores y compartir mejor la información entre las herramientas. Por ejemplo, una herramienta de gestión de calidad puede integrarse mejor con un sistema de gestión de cambios para registrar los defectos de software y realizar su seguimiento. **[1]**

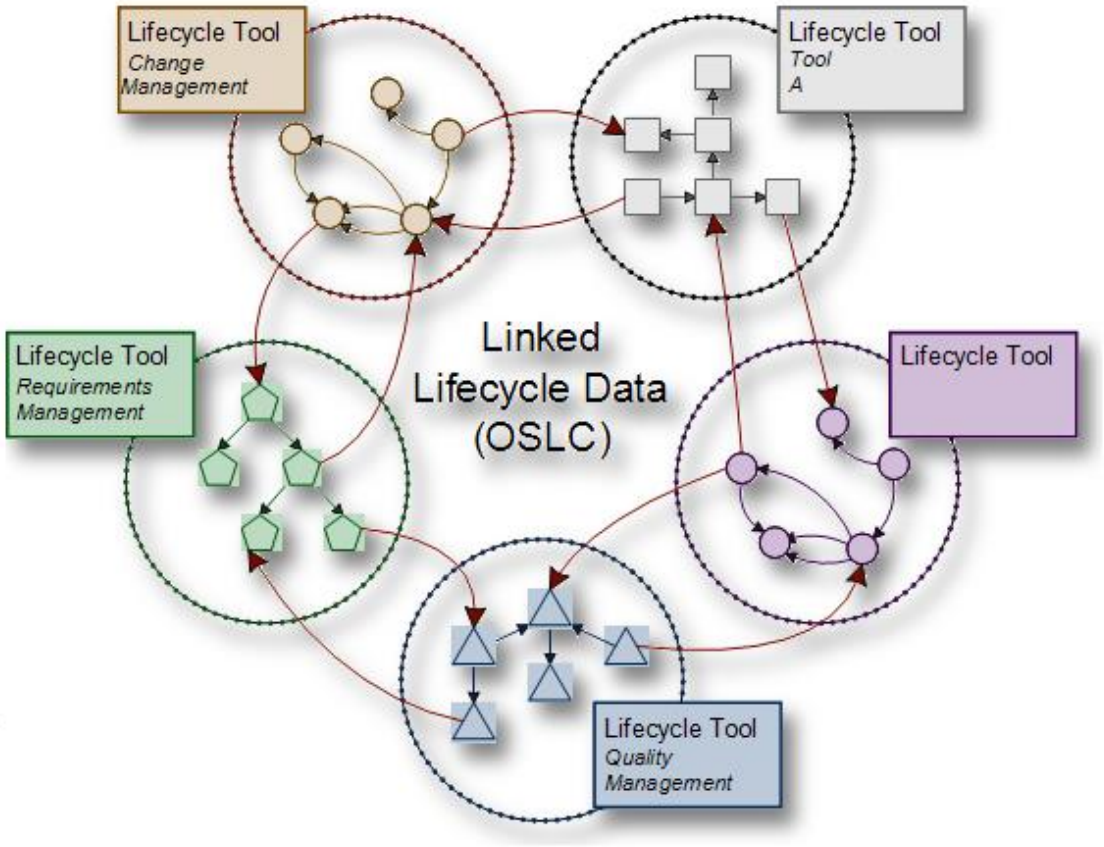


Ilustración 2: Interoperabilidad entre herramientas

Los recursos OSLC están pensados para ser consumidos por OSLC. Cada estructura de objeto incluye el objeto de negocio primario y los objetos de negocio hijo utilizados por una aplicación.

Cada tipo de recurso OSLC tiene propiedades que son predicados RDF que pueden pertenecer a una especificación de vocabulario que corresponde al espacio de nombre de las propiedades. Los atributos pueden correlacionarse con las propiedades definidas en las especificaciones de vocabulario.

Cada recurso tiene un nombre, una estructura de objeto asociada consumida por OSLC y el URI de espacio de nombre predeterminado del recurso. Para cada atributo que deba integrarse se pueden utilizar los valores RDF predeterminados o especificar nuestros propios valores. *[2]*

1.2. Quartus II

Quartus II es una herramienta de software producido por Altera para el análisis y la síntesis de los diseños en HDL. Permite al desarrollador compilar sus diseños, realizar análisis temporales, examinar diagramas RTL y configurar el dispositivo de destino con el programador.[3]

Tiene muchas ventajas, pero entre todas ellas a nosotros nos interesa que posee un compilador capaz de recibir como entrada archivos con circuitos esquemáticos o archivos VHDL o VERILOG.

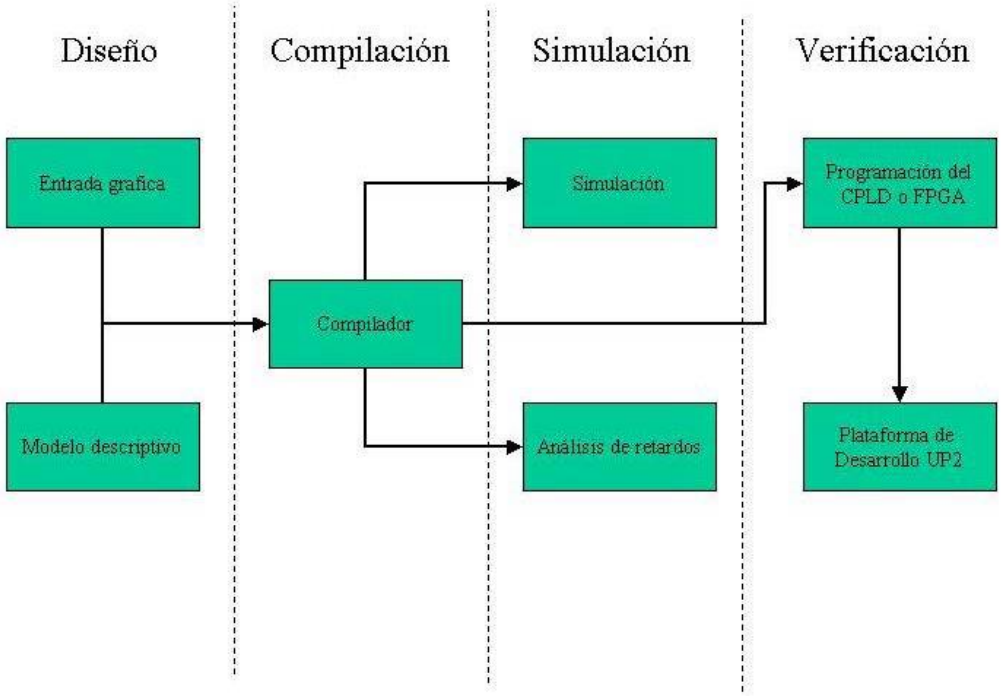


Ilustración 3: Etapas del proceso de diseño en la plataforma ALTERA

1.3. Motivación y objetivos

Ahora que hemos visto la definición y lo que hacen los dos pilares fundamentales del proyecto es más fácil explicar lo que buscamos al hacerlo.

El objetivo principal de este proyecto es poder unificar herramientas para la optimización del trabajo. Para ello vamos a crear una herramienta que funcione como adaptador de OSLC, en este caso para Quartus II.

Como hemos dicho Quartus II puede hacer muchas cosas, pero nos vamos a centrar en el modelado de diagramas de bloques ya que necesitamos principalmente analizar la codificación de VHDL de los diagramas.

Nuestro primer objetivo es ser capaces de representar y reutilizar modelos físicos en VHDL. Esto lo haremos transformando a SRL y utilizando las APIs de CAKE.

Queremos que se pueda usar y sirva para integrar todo lo que se haga con Quartus con otras herramientas compatibles y así el intercambio de información sea mucho más fácil y efectivo.

Con esto pretendemos ahorrar tiempo y dinero a las empresas, tanto usuarios de herramientas, como a compradores y vendedores de estas, ya que está pensado de forma genérica para que todo el mundo lo pueda usar y no de forma específica, lo que facilita mucho la integración entre herramientas.

Ilustración 4: Integración

Hay un gran beneficio comercial tanto para los productores como para los consumidores de las API de OSLC. Los productores pueden crear una API estándar sobre una variedad de herramientas, las cuales pueden construir y pueden crear integraciones de alta, muy alta calidad, ya sean integraciones del IDE o integraciones con dispositivos móviles para que no accedan los programadores.

Los consumidores de las API, obtienen una experiencia de usuario consistente en todas sus tecnologías ALM, y obtienen soporte de herramientas de última generación, incluso para sistemas heredados de los que aún no pueden migrar.

2. ESTRUCTURA DEL DOCUMENTO

Como se puede apreciar durante todo la memoria, el documento se ha fragmentado en diferentes puntos para conseguir que sea lo más clara y concisa posible.

- *Introducción*

En este apartado se intenta introducir al lector en el proyecto, para que entienda de que trata y que se va a desarrollar en él.

- *Estado del arte*

En esta sección se realiza un estudio sobre como se ha desarrollado tecnológicamente todo en lo que se basa nuestro proyecto, la herramienta se basa en interoperabilidad, integración, OSLC… y se pretende mostrar como ha ido evolucionando y el por qué es tan importante.

- *Análisis del Sistema*

En el análisis se definen los requisitos del sistema y casos de uso.

- *Diseño de la herramienta*

En este apartado se muestran los componentes y como será el funcionamiento del sistema, sin entrar en la implementación.

- *Implementación y pruebas*

En esta parte se explica la implementación del sistema. Se mostrarán las pruebas realizadas y necesarias para comprobar el funcionamiento de la herramienta.

- *Planificación y presupuesto*

En estos apartados se muestra la planificación realizada para la realización del trabajo, y el presupuesto de éste.

- *Conclusiones y trabajo futuro*

Esta sección contiene las conclusiones sobre el proyecto y una reflexión para trabajos futuros.

3. ESTADO DEL ARTE

3.1. Industria 4.0

"La Industria 4.0, también llamada industria inteligente, se considera la cuarta revolución industrial y busca transformar a la empresa en una organización inteligente para conseguir los mejores resultados de negocio." [4]

La transformación digital ha conseguido que las diferentes disciplinas de ingeniería convivan y colaboren para crear productos y servicios más complejos gracias a algún tipo de software como CPS [5] [6] (Smart grids). [7]

Tanto el diseño como la implementación del sistema deben tener mecanismos de interoperabilidad e integración suficientes para que las máquinas y las personar puedan trabajar conjuntamente para conseguir cosas de mayor nivel, de forma más óptima, mejorando la seguridad, la reducción del tiempo o incluso el coste.

Antes de todo esto, era necesario aplicar los principios de ingeniería de las disciplinas asociadas con el sector en concreto con el que se trabajaba, ferroviario, automoción…
Ayudando con otras disciplinas como la ingeniería de software. Los procesos técnicos necesitaban el apoyo de aplicaciones software, las cuales ofrecían determinadas funcionalidades. IBM Doors o Matlab Simulink son ejemplos de estas aplicaciones diseñadas para un método de ingeniería en particular.

A través de documentos o del mismo correo electrónico era como habitualmente se realizaban las comunicaciones entre los distintos tipo de ingeniería. De este modo, en la ingeniería no había colaboración y comunicación entre herramientas.

Todo esto supone una situación con muchas desventajas, tanto técnicas como de ingeniería. Entre ellas cabe destacas problemas y situaciones como la de vendor lock-in en la que el uso de aplicaciones de un vendedor no ha sido diseñada para integrarla y que tenga interoperabilidad, o la dificultad para reutilizar el conocimiento ya utilizado por otros equipos de ingeniería.

Se han elaborado especificaciones y estándares con el objetivo de mejorar la ingeniería. Estas especificaciones y estándares son relativos a la necesidad de integrar distintas herramientas. Actualmente, gracias a estos estándares, debidos a la unión entre industria y academia, se puede dar soporte a los artefactos que se generan durante el ciclo de vida de desarrollo. Un buen ejemplo de esto es OSLC (OASIS).

Utilizando este ejemplo, podemos ver que ya se han desarrollado proyectos de interoperabilidad de modelos físicos usando un adaptador de OSLC para Modelica. [8]

3.2. Comunicación y OSLC

La comunicación es la acción consciente de intercambiar información entre dos o más participantes con el fin de transmitir o recibir información u opiniones distintas.

La mayoría de las especies tiene sus medios de comunicación, pero la humana es la que ha logrado desarrollarla de una forma mucho más rápida.

A través de los siglos los sistemas de comunicación ha variado mucho. Siempre hemos usado todo lo que encontrábamos para crear sistemas comunicativos para poder interactuar y poder avanzar como sociedad. Sin comunicación no podríamos haber alcanzado la primera posición respecto a supervivencia y desarrollo, hasta el punto en el que ahora nos encontramos de desarrollo tecnológico.

- Pinturas rupestres (Atapuerca, 30.000 a.C.)
- Pictogramas (China y Egipto, 5.000 a.C.)
- Papel (China, 105 d.C.)
- Código Morse (Estados Unidos, 1835 d.C.)
- Teléfono (Estados Unidos, 1876 d.C.)
- ARPANET (Estados Unidos, 1969 d.C.)

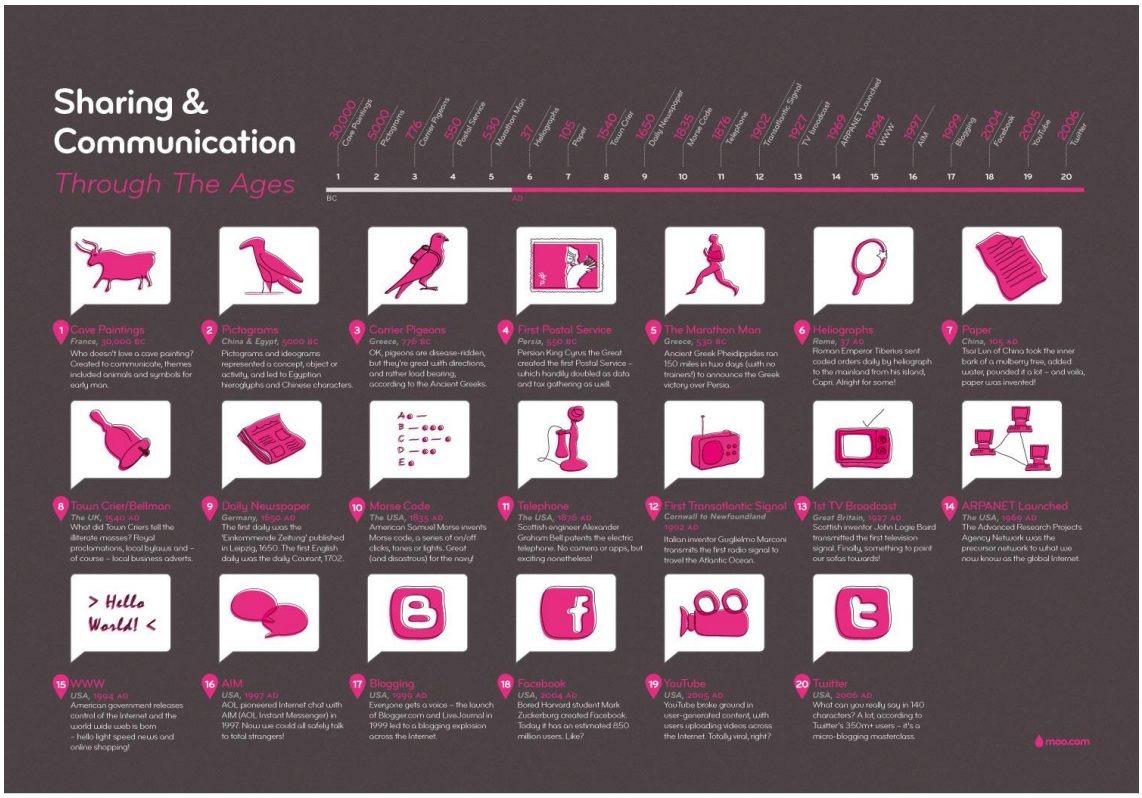


Ilustración 5: Tipos de comunicación

Por todos es sabido la importancia de encontrar un lenguaje común para favorecer el intercambio de información y así avanzar como sociedad. Como ejemplos de estos lenguajes tenemos entre otros:

- Código Morse
- Código Binario
- Código de Barras
- Señalización vial

De este espíritu por encontrar un lenguaje común y del contexto de reutilización de artefactos en ingeniería nace en 2008 OSLC (Open Services for Lifecycle Collaboration) con el fin de ser un conjunto de soluciones para reducir la complejidad a la hora de integrar diversas herramientas de software.

Para ello, OSLC ofrece dos técnicas principales para integrar herramientas: "Vinculación de datos a través de HTTP" y "Vinculación de datos a través de la interfaz de usuario HTML". Ambas técnicas se basan en la base HTTP y RDF de OSLC.

- Vinculación de datos a través de HTTP. OSLC especifica un protocolo de herramienta común para crear, recuperar, actualizar y eliminar datos de ciclo de vida (CRUD) basados en estándares de Internet como HTTP y RDF utilizando el modelo de datos vinculados. Este protocolo puede ser utilizado por cualquier herramienta u otro cliente programático para hablar con cualquier otra herramienta que implemente las especificaciones. La vinculación se logra integrando la URL HTTP de un recurso en la representación de otro. http://www.sparxsystems.com.ar/EAUserGuide/index.html?oslc_requirements_management.htm

- Vinculación de datos a través de la interfaz de usuario HTML. OSLC especifica un protocolo que permite que una herramienta u otro cliente haga que se muestre un fragmento de la interfaz de usuario web de otra herramienta, permitiendo que un usuario humano se vincule a un recurso nuevo o existente en la otra herramienta o vea una vista previa de información sobre un recurso en otra herramienta. Esto permite que una herramienta u otro cliente explote la interfaz de usuario existente y la lógica empresarial en otras herramientas al integrar información y pasos de proceso. En algunas circunstancias, esto es más eficiente y ofrece más funciones de usuario que implementar una nueva interfaz de usuario y luego integrarla a través de un protocolo HTTP CRUD.

3.3. Interoperabilidad

La interoperabilidad es la capacidad de un sistema de información, de comunicarse y compartir documentos, datos y objetos digitales de forma efectiva mediante una conexión automática, transparente y libre. La interfaz del sistema propio no se deja de usar en ningún momento. [10]

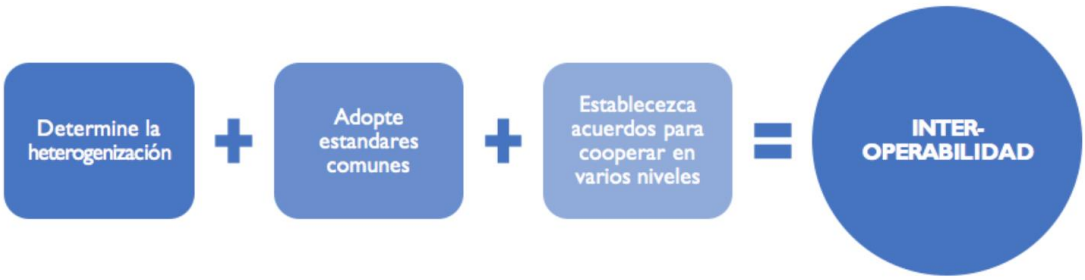


Ilustración 6: Interoperabilidad

Para integrar información de distintas fuentes, se utilizan herramientas PLM o ALM, consiguen hacer esto consultando la información mediante un protocolo de comunicación común.
Todavía hay muchos artefactos para los que no existe un esquema definido, como por ejemplo para el nuestro, modelos hardware.

Debido a esto, existe una tendencia hacia la interoperabilidad, para conseguir crear un "almacén" de datos en el cual puedan estar representados los metadatos propios de un artefacto.

Todo esto, utilizando un modelo RDF y un protocolo de comunicación (HTTP REST) permitiría ofrecer toda esta información mediante OSLC. De esta forma se conformaría un modelo [11] que ayudaría a otros procesos de desarrollo como las líneas de producto [12]. Sin embargo, debido a las capacidades de la representación de información en RDF [13], la base de OSLC queda ensombrecida, ya que RDF tiene restricciones en cierto aspectos, además de problemas relacionados con la cosificación de información [14].

3.3.1. Como alcanzar la interoperabilidad

La interoperabilidad puede conseguirse mediante distintos caminos complementarios:[15]

- La adopción de estándares abiertos: Provienen de organizaciones como ISO o ECMA y deben tener ciertas características:

1. Disponible públicamente sin costo o con unos precios razonables para su adopción y puesta en práctica por cualquier parte interesada.

2. Todo lo relativo a patentes que sea de necesidad para la implementación de estándares abiertos ha de estar a disposición del total de implementadores por parte de quienes estén al cargo del desarrollo de la especificación en términos no discriminatorios (RAND) y razonables con o sin necesidad de abonar unas tarifas razonables.

3. Mantenida, desarrollada, ratificada, o aprobada mediante consenso por el mercado abierto a todos los participantes cualificados e interesados, en una organización creadora de estándares.

4. Para permitir un entendimiento del alcance estándar y del objetivo tiene que ser publicada sin restricciones.

5. Adopción de "recomendaciones": Se incluyen especificaciones técnicas de organismos de la industria tales como el W3C. No son "estándares" en sí, pero sí especificaciones en que los fabricantes acuerdan dar soporte. Estas especificaciones pueden adjuntarse a propiedad intelectual específica.

6. Adopción de normas "propietarias": estas son especificaciones técnicas que son desarrolladas y mantenidas por una única entidad o por un grupo privado y cerrado de entidades que cooperan entre sí, y que están normalmente disponibles mediante su publicación y ampliamente licencia- das bajo condiciones

comercialmente razonables, de modo que puedan ser ampliamente adoptadas por todo el mercado (por ejemplo, Adobe PDF, Java, APIs Win 32).

7. Formatos publicados, APIs o protocolos: estas son especificaciones publicadas que permiten la interoperabilidad con varias plataformas y aplicaciones. Pueden ser formatos de archivo que permiten la interoperabilidad a nivel de intercambio de ficheros (como por ejemplo el esquema XML para aplicaciones como Microsoft Office), APIs que permiten la interoperabilidad a nivel de programa entre una aplicación y una plataforma subyacente, y los protocolos que definen la manera mediante la cual el software "habla" directamente a través del “cable”.

8. Publicación y licenciamiento de tecnologías y propiedad intelectual relacionada: estas son especificaciones, procesos, o interfaces que son publicados y ampliamente licenciados (por ejemplo, el programa de licencias de patentes de IBM).

9. Esfuerzos de colaboración en la industria orientados por voluntarios; estos son esfuerzos entre *partners* y competidores para solucionar problemas relacionados con la interoperabilidad, con la intención de satisfacer al cliente y la demanda del mercado (por ejemplo, el acuerdo de Interoperabilidad Sun-Microsoft).

3.3.2. Problemas con la interoperabilidad

Existen dos principales problemas por los que no se logra la interoperabilidad en los sistemas de información: [15]

 Aspectos sociales: la inercia social respecto al cambio y al modo del trabajo de los sistemas de formación existentes respecto al nuevo modo de trabajo es enorme, lo cual retrasaría cualquier implementación de un sistema de información interoperable y homogéneo.

 Aspectos técnicos: Pocas empresas ven realmente la necesidad de realizar este proceso, ya que los costos asociados son bastante altos y pocas empresas ven realmente la necesidad de realizar este proceso.

A parte hay otros problemas como el no poseer un lenguaje o una visión común general de aplicación; No conseguir soluciones necesitadas por la organización; Deficiencias en las vías para comunicar; Los sistemas de las organización no tienen capacidad de adaptación frente a posibles cambios.

3.4. Integración

Una herramienta de integración tiene muchos beneficios debido a que con ella se pueden reutilizar la mayoría de las aplicaciones de una empresa y reemplazar las que ya no tienen uso.
De esta forma es posible no tener que reemplazar todo el sistema, lo que requiere mucho más presupuesto en licencias e inversión.

La mejor forma de alcanzar objetivos reales de integración de datos es mediante aplicaciones compuestas, estas aportan escalabilidad a los sistemas heredados. Lo bueno de esto es que no altera al resto de aplicaciones que se estén usando en las empresas.
Y es que en un entorno como en el que nos encontramos, donde hay un gran diversidad de canales, de servicios en la nube… es un hecho que la integración continua siendo un reto para las empresas.

Actualmente, la cantidad de información disponible no tiene precedentes, por eso muchas organizaciones están aprovechando toda esta información para cambiar sus procesos de negocio. Las empresas usan herramientas de analítica, las posibilidades de interacción que ofrecen las redes, es decir, las que saben aprovechar esta información tienen unas ventajas frente a la competencia importantes.

De todas formas todo esto se basa en una integración escalable, fiable y flexible. Es necesario aunar toda la nueva información con los datos ya almacenados en las

herramientas que ya se están utilizando y de esta forma, como ya hemos dicho cambiar los procesos de negocio.

Para la adopción de nuevos modelos cubriendo de esta forma los gaps tecnológicos referentes a los sistemas operaciones, se necesita un software de integración. Con esto se facilita su evolución continua y el horizonte temporal de estos se alarga.
Las herramientas de integración son fundamentales, ya que si falta una herramienta específica es necesario integrar sistema a sistema. Por ejemplo si tienes 5 sistemas, cada uno de ellos tiene que conectarse a los otros 4, y cuando haya cambios en alguno de ellos, se deberían rehacer esas 4 conexiones. [16]

4. ANÁLISIS DEL SISTEMA

En este apartado se va implementar la solución al problema planteado. Aquí se podrán ver los requisitos y necesidades de la herramienta a desarrollar.

A parte se verán los casos de uso con el fin de analizar el sistema.

4.1. Planteamiento del problema

Como hemos visto en apartados anteriores, vamos a hacer una herramienta para poder enlazar y compartir datos utilizando la integración de Open Services for Lifecycle Collaboration.

Lo que queremos haciendo esto es conseguir una integración hardware mediante los diagramas de bloques de Quartus II, transformar estos datos a SRL y de esta forma meterlos en el repositorio de CAKE para conseguir la interoperabilidad y reutilización de los datos.

La idea es crear una aplicación para poder unificar y compartir datos más fácilmente mediante OSLC. Elegimos Quartus II ya que es un programa muy usado en electrónica y programa en VHDL, algo importante ya que lo buscábamos a propósito para facilitar la tarea de la integración en OSLC.

El proveedor pone sus datos de recurso a disposición de la aplicación del consumidor a través de contenedores, conocidos como proveedores de servicio. Con los datos de recursos disponibles, nuestra aplicación puede crear enlaces entre sus datos y los datos de recursos de la aplicación de proveedor. Puede configurar y habilitar cualquier aplicación de su producto para que actúe como una aplicación de consumidor de OSLC.

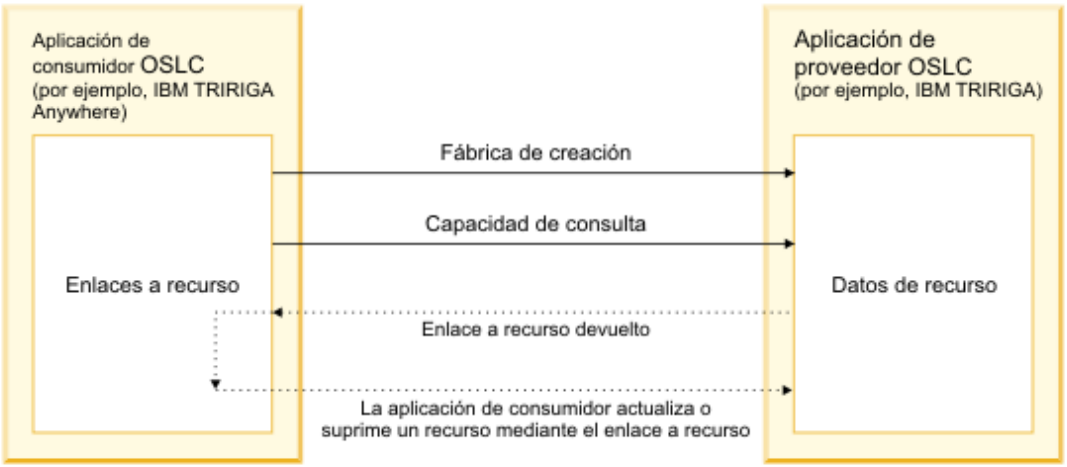


Ilustración 7: Ejemplo tipo aplicación OSLC

4.2. Requisitos

Los requisitos son fundamentales para elaborar un proyecto. El propósito de la gestión de requerimientos es asegurar que el proyecto cumple con las expectativas de sus clientes y de sus interesados.

Estas especificaciones se van a recoger en tablas con este formato:

INDENTIFICADOR : R – XX(1)	
TÍTULO(2)	
PRIORIDAD(3)	
DESCRIPCIÓN(4)	

Tabla 1: Ejemplo tabla requisitos

(1) Identificador: Tiene un nombre único y permite identificar el requisito.

(2) Título: Nombre del requisito.

(3) Prioridad: Es la importancia que le da el cliente al requisito. Puede ser alta, media o baja.

(4) Descripción: Explicación del requisito.

INDENTIFICADOR : R – 01	
AUTOR	Sara Martín Fraile
PRIORIDAD	Alta
DESCRIPCIÓN	El sistema leerá archivos VHDL

Tabla 2: Requisito funcional R-01

INDENTIFICADOR : R – 02	
AUTOR	Sara Martín Fraile
PRIORIDAD	Alta
DESCRIPCIÓN	El sistema transformará VHDL al lenguaje SRL

Tabla 3: Requisito funcional R-02

INDENTIFICADOR : R – 03	
AUTOR	Sara Martín Fraile
PRIORIDAD	Alta
DESCRIPCIÓN	El sistema pasará VHDL por contenido

Tabla 4: Requisito funcional R-03

INDENTIFICADOR : R – 04	
AUTOR	Sara Martín Fraile
PRIORIDAD	Alta
DESCRIPCIÓN	Se utilizará Quartus II para hacer los diagramas de bloques

Tabla 5: Requisito funcional R-04

INDENTIFICADOR : R – 05	
AUTOR	Sara Martín Fraile
PRIORIDAD	Alta
DESCRIPCIÓN	La transformación de VHDL a SRL estará construida en .NET

Tabla 6: Requisito funcional R-05

INDENTIFICADOR : R – 06	
AUTOR	Sara Martín Fraile
PRIORIDAD	Alta
DESCRIPCIÓN	El sistema transformará VHDL a SRL usando C++

Tabla 7: Requisito funcional R-06

INDENTIFICADOR : R – 07	
AUTOR	Sara Martín Fraile
PRIORIDAD	Alta
DESCRIPCIÓN	La API utilizada será CAKE

Tabla 8: Requisito funcional R-07

INDENTIFICADOR : R – 08	
AUTOR	Sara Martín Fraile
PRIORIDAD	Alta
DESCRIPCIÓN	El sistema usará un servicio RSHP

Tabla 9: Requisito funcional R-08

INDENTIFICADOR : R – 09	
AUTOR	Sara Martín Fraile
PRIORIDAD	Alta
DESCRIPCIÓN	El sistema usará KM como repositorio para almacenar toda la información y poderla reutilizar

Tabla 10: Requisito funcional R-09

4.3.Casos de uso

Analizados ya los requisitos, a continuación analizaremos los casos de uso del sistema. De esta forma podremos estudiar la secuencia de accione. En el siguiente diagrama se representan los posibles usos con las acciones que se permiten al usuario.

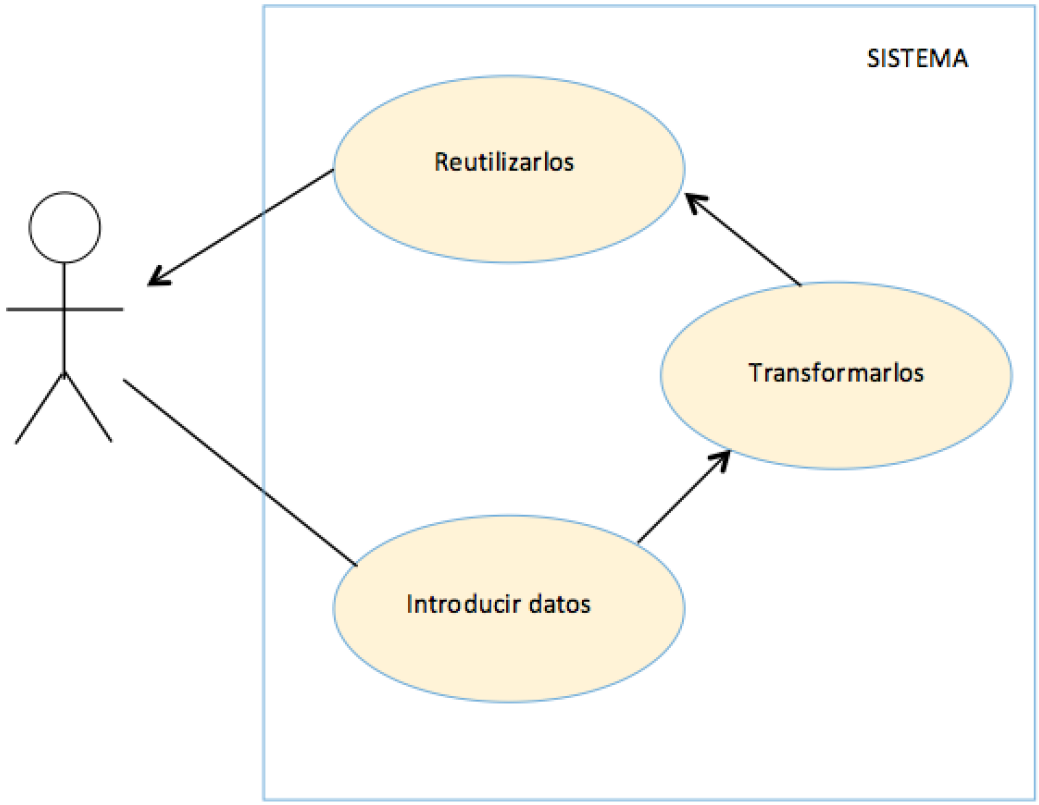


Ilustración 8: Casos de uso

Como se puede ver en la ilustración 8, el sistema es el que se encarga de hacer todos los pasos para conseguir la reutilización de los datos.

5. DISEÑO DEL SISTEMA

En este apartado se va a mostrar el diseño del sistema. Se divide en varios apartados en los que se pretende mostrar todo el pensamiento llevado a cabo para conseguir el diseño de la herramienta. Se van a describir tanto los lenguajes de programación usados como los programas utilizados y el mapeo entre las entidades de VHDL y SRL para llegar a la solución buscada.

5.1. Diseño abstracto

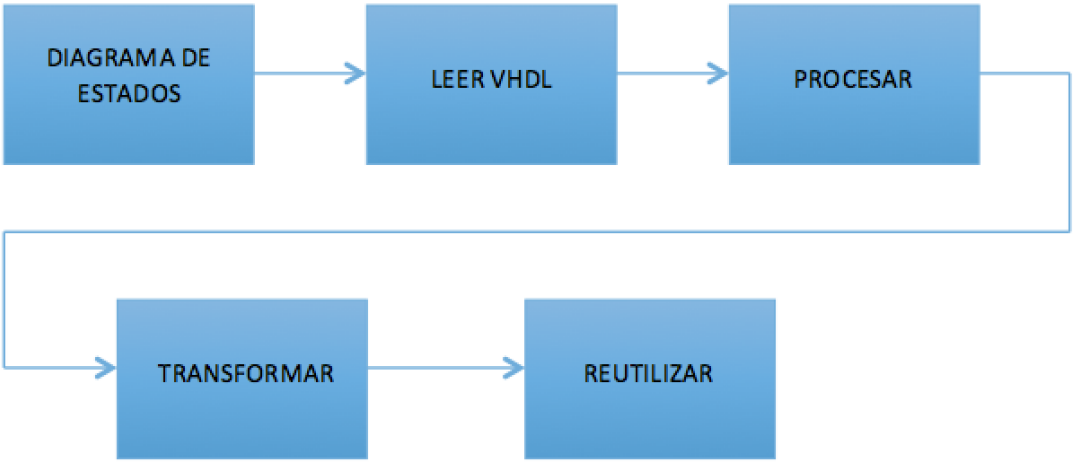


Ilustración 9: Diseño abstracto del sistema

En el diagrama anterior, podemos ver el diseño abstracto del proyecto. Nuestro proyecto se basa en leer en VHDL los diagramas de estados realizados en Quartus II, procesar esa información y analizarla para poderla transformar y por fin conseguir la interoperabilidad reutilizando esta información.

5.2.Tecnología y entorno de desarrollo

En el siguiente diagrama podemos ver la tecnología utilizada para el diseño de la herramienta:

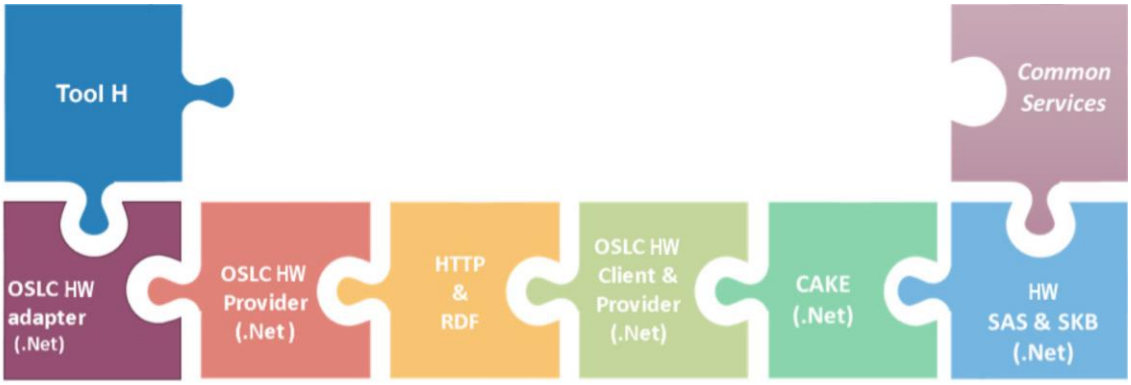
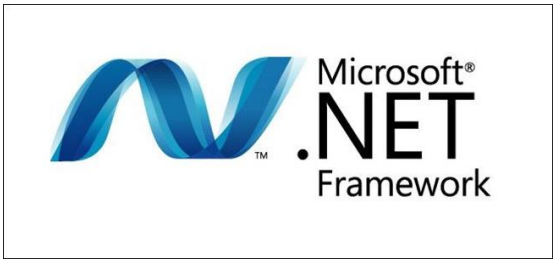


Ilustración 10: Tecnología de la herramienta

A continuación se van explicar las herramientas y frameworks principales utilizados para el desarrollo del proyecto:

5.2.1. .NET



Es un framework de Microsoft que hace un énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permite un rápido desarrollo de

Ilustración 11: Logo Microsoft .NET

aplicaciones. Basada en ella, la empresa intenta desarrollar una estrategia horizontal que integre todos sus productos, desde el sistema operativo hasta las herramientas de mercado. [17]

Nuestra herramienta esta construida sobre .NET usando C++. Con esto pretendemos transformar lo procesado al leer el archivo VHDL en SRL (OSLC) para el repositorio de CAKE.

5.2.2. API CAKE

API significa "Interfaz de programa de aplicación", que indica un conjunto de rutinas, protocolos y herramientas para crear aplicaciones de software. Se utilizan para acceder a información de bases de datos o realizar ciertas acciones fuera de una interfaz de usuario.

Se trata de una estructura que sirve de base a los programadores para que puedan crear aplicaciones Web. El objetivo principal es trabajar de forma rápida y estructurada..

CakePHP es un marco de desarrollo (framework) rápido para PHP, de código abierto.

Esta es una lista breve con las características de CakePHP: [27]

- Comunidad activa y amistosa
- Licencia flexible
- Compatible con PHP4 y PHP5
- CRUD integrado para la interacción con la base de datos
- Soporte de aplicación [scaffolding]
- Generación de código
- Arquitectura Modelo Vista Controlador (MVC)
- Despachador de peticiones [dispatcher], con URLs y rutas personalizadas y limpias
- Validación integrada
- Plantillas rápidas y flexibles (sintaxis de PHP, con ayudantes[helpers])
- Ayudantes para AJAX, Javascript, formularios HTML y más
- Componentes de Email, Cookie, Seguridad, Sesión y Manejo de solicitudes
- Listas de control de acceso flexibles
- Limpieza de datos
- Caché flexible
- Localización
- Funciona en cualquier subdirectorio del sitio web, con poca o ninguna configuración de Apache

5.2.3. Knowledge Manager

El conocimiento es uno de los activos más valiosos en su organización. El impulso clave del éxito en cualquier proyecto de sistema y software es reutilizar los activos de conocimiento, como el conocimiento explícito y tácito de los ingenieros, y las pautas que definen el conocimiento de la organización.

Ilustración 12: Knowledge Manager

Por lo tanto, el conocimiento debe recopilarse de diferentes fuentes, almacenarse en repositorios seguros y acceder al personal designado en el momento adecuado.

KM - Knowledge Manager permite administrar el conocimiento desde el punto de vista de ingeniería de sistemas y almacenar información valiosa de requisitos, modelos, arquitecturas de sistemas y otros documentos en una Base de Conocimientos de Sistemas común. [19]

5.3. Transformación de VHDL a SRL

5.3.1. Metamodelo VHDL

Para empezar se va a dar un poco de información sobre VHDL:

VHDL es un lenguaje de especificación definido por el IEEE (Institute of Electrical and Electronics Engineers) (ANSI/IEEE 1076-1993) es utilizado para

lustración 13: Logo VHDL

describir circuitos digitales y para la automatización de diseño electrónico. VHDL es un acrónimo que viene de la combinación de dos acrónimos: VHSIC (Very High Speed Integrated Circuit) y HDL (Hardware Description Language). Aunque puede ser usado de forma general para describir cualquier circuito digital se usa principalmente para programar PLD (Programable Logic Device - Dispositivo Lógico Programable), FPGA

(Field Programmable Gate Array), ASIC y similares. Es un lenguaje similar a ADHL o Verilog, Estos lenguajes presentan un mismo objetivo y se diferencia del clásico C (o cualquiera parecido a este) por ser un lenguaje paralelo no secuencial. [20]

Un FPGA no es como un microcontrolador, en realidad es un conjunto masivo de celdas o bloques lógicos programables. Estas celdas son programadas individualmente para convertirse en pequeños bloques de construcción. Pueden ser compuertas simples (AND, OR y NOT, etc) o flip-flops. Así que la diferencia entre una FPGA y un micro es que nosotros programamos cada celda para funcionar como un bloque de lógica. Esto significa que, debido a que cada bloque es tan independiente, todos están operando al mismo tiempo. A diferencia de un micro donde cada línea de código se procesa a su vez.

Un programa realizado en VHDL puede parecer como un programa de computadora clásico (secuencial) y puede procesar los datos en los cambios de flanco del clock, pero hay que recordar que todo está funcionando a la vez (paralelo). [21]

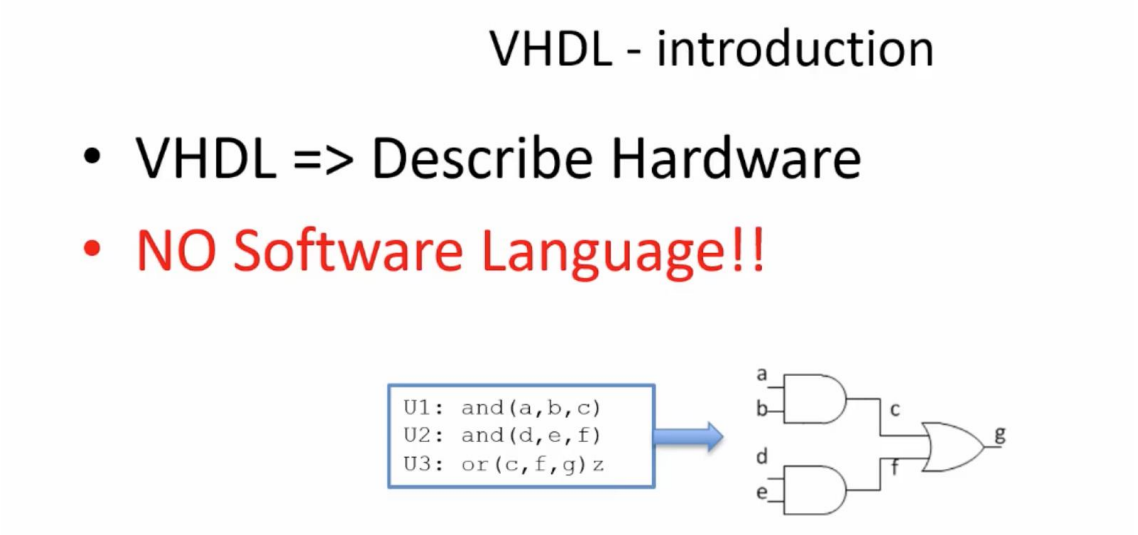


Ilustración 14: VHDL - introduction

VHDL y su hermano Verilog se han convertido en la herramienta standard a la hora de diseñar circuitos digitales complejos.

5.3.1.1. Entidades y arquitecturas

Para determinar un circuito, VHDL dispone de las construcciones ENTITY y ARCHITECTURE.(www.uc3m.es)

- Una entidad describe las entradas y salidas de la interfaz de un circuito.

- Una arquitectura permite describir la funcionalidad que realiza el circuito y siempre está asociada con una entidad definida.

La entidad es el bloque de construcción básico de VHDL.

Todos los diseños VHDL son implementados por entidades. Se puede imaginar la entidad como un cuadro negro con puertos de entrada y salida. Las entidades permiten crear una jerarquía en el diseño.

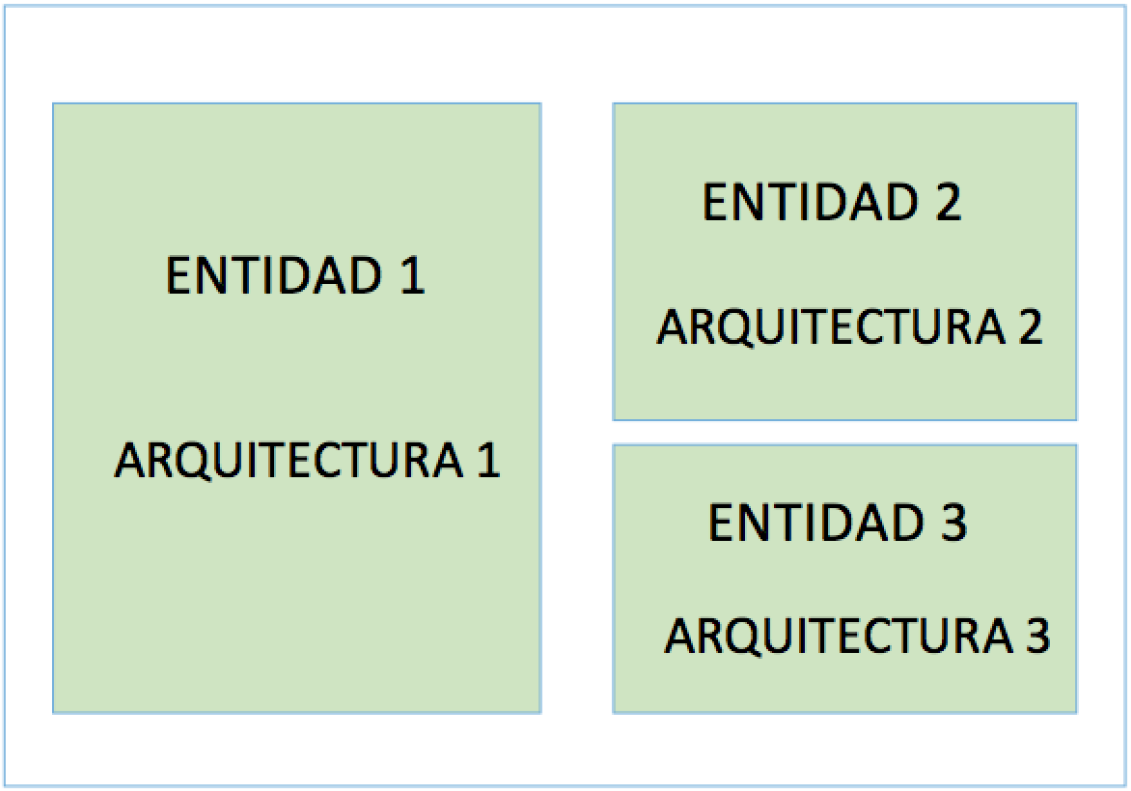


Ilustración 15: Jerarquía VHDL

En este ejemplo se puede ver bien a que nos estamos refiriendo:

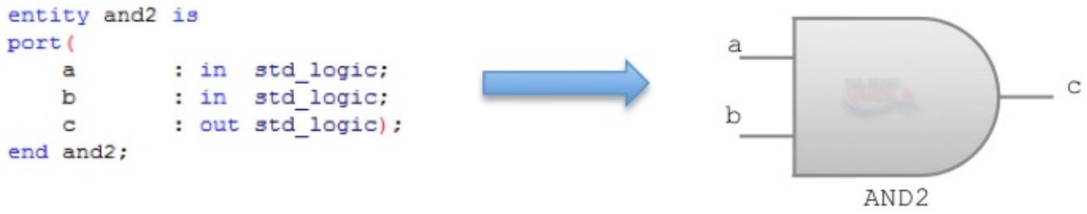


Ilustración 16: Ejem. 1 Puerta lógicas con VHDL (Entidad)

La sintaxis de la entidad es la palabra clave " entity ", seguida del nombre de la entidad y la palabra clave " is " y " port ".

Dentro del paréntesis está la declaración de los puertos. Primero está el nombre del puerto, luego la dirección del puerto, seguido del tipo de puerto. La declaración de entidad termina con la palabra clave " end " seguida del nombre de la entidad.

La declaración de arquitectura describe la funcionalidad subyacente de la entidad. La arquitectura siempre está relacionada con una entidad y describe el comportamiento de esa entidad.

Por ejemplo, la entidad y2 se describe mediante la arquitectura denominada " and2_a "

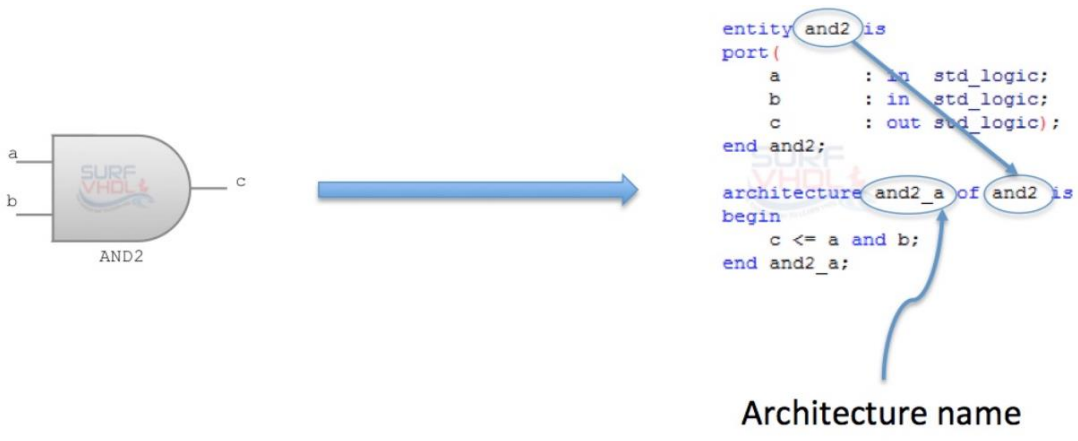


Ilustración 17: Ejm. 2 Puertas lógicas con VHDL (Arquitectura)

La arquitectura contiene sentencias para describir que hace el circuito. Estas sentencias pueden ser de dos tipos dependiendo de su ejecución en la simulación.

• Sentencias concurrentes: si se ejecutan en paralelo en el simulador, no importa en el orden en el que se escriban, como se puede ver en el siguiente ejemplo: [29]

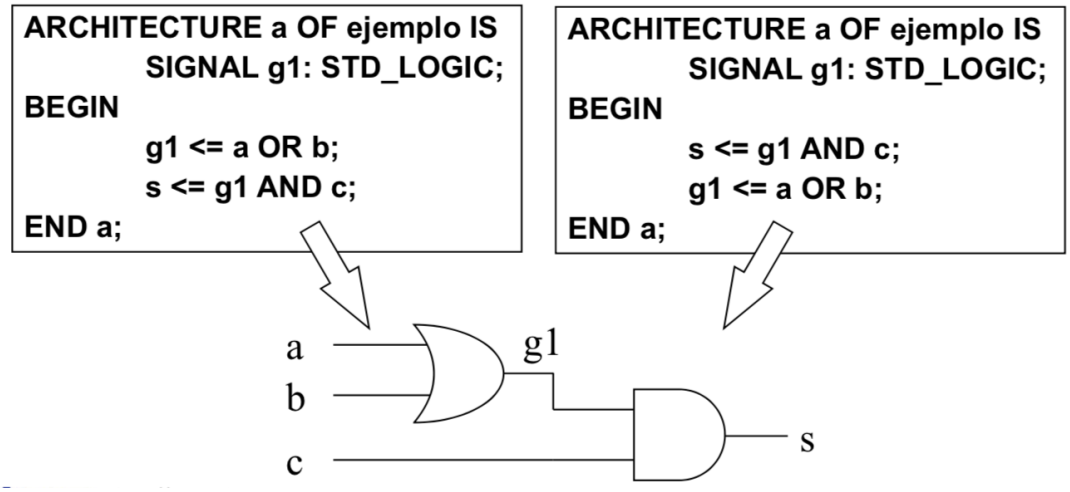


Ilustración 18: Sentencias concurrentes VHDL

De esta forma se escriben las sentencias concurrentes:

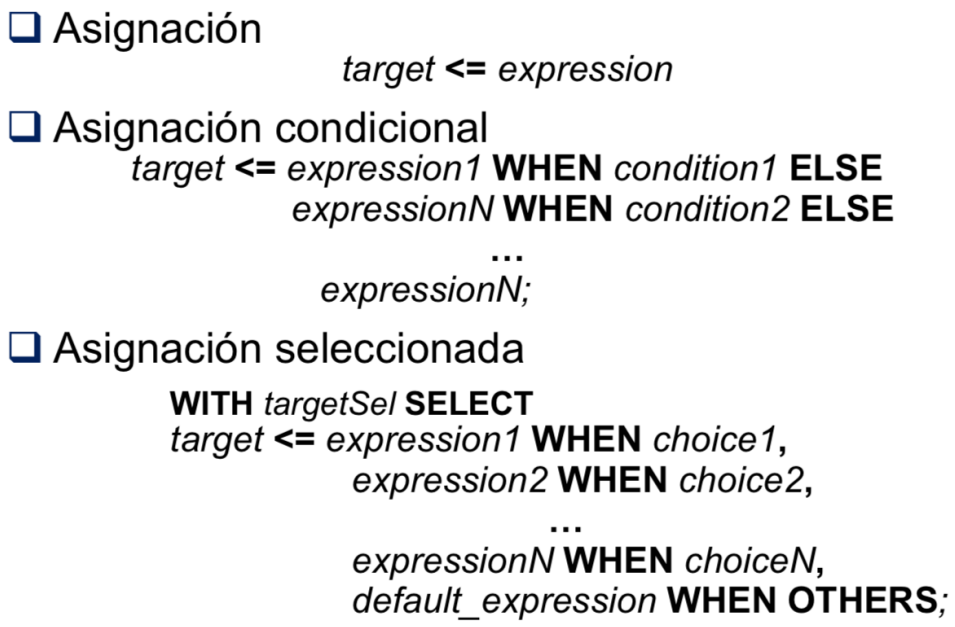


Ilustración 19: Como se escriben las sentencias concurrentes

• Sentencias secuenciales si se ejecutan de manera secuencial. No confundir el término secuencial cuando se usa para describir a un circuito (circuitos secuenciales tienen elementos de memoria). Se ejecutan una detrás de otra, a parte no tiene un efecto inmediato en la simulación si no que al final se tienen en cuenta todas las sentencias.

Un ejemplo para este tipo de sentencias, es la sentencia IF:

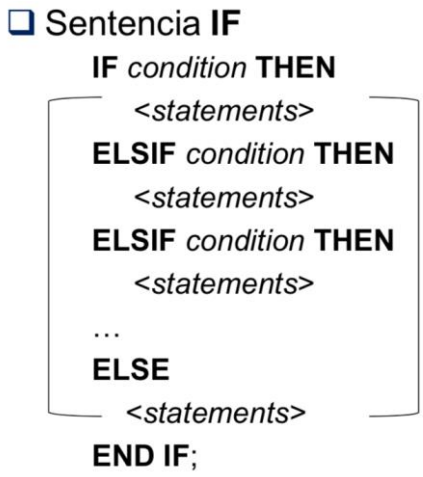


Ilustración 20: Sentencia IF - Sentencia secuencial

5.3.2. Metamodelo SRL

Como resultado de estudios previos, se ha propuesto una evolución de la especificación preliminar de OSLC KM (Gestión del conocimiento) para definir una forma común para cualquier tipo de artefacto del sistema (considerado un activo de conocimiento) que debe representarse, almacenarse, compartirse o intercambiarse entre herramientas en un proceso de desarrollo. Por otro lado, la iniciativa OSLC está adquiriendo un fuerte compromiso para aplicar los principios de Linked Data, RDF y REST para impulsar la interoperabilidad.

Las especificaciones o, más precisamente, las formas de datos, ya se han definido para modelar metadatos y contenido de requisitos, activos, casos de prueba, cambios y métricas de estimación y medición. Del mismo modo, el grupo OMG está trabajando en la especificación OSLC-MBSE para promover modelos de sistemas a datos vinculados. Sin embargo, todavía hay algunos artefactos para los cuales no hay forma, como un

elemento de un vocabulario, un patrón de requisitos o un modelo de sistema dinámico. Debido a este hecho, es difícil trazar una estrategia común para la gestión del conocimiento. Además, algunos servicios transversales como la indexación y los procesos de recuperación se delegan en herramientas de terceros, lo que impide la implementación de uno de los pilares de la gestión del conocimiento para la reutilización del software: la selección. Por lo tanto, presentamos una forma de datos para cualquier artefacto generado durante el ciclo de vida del desarrollo.

1) SRL (lenguaje de representación del sistema): el lenguaje para representar los metadatos y el contenido de cualquier artefacto.

2) Knowledge MANAGER: la herramienta básica que proporciona los servicios necesarios para el repositorio de conocimiento de software.

3) Forma de datos RDF e interfaz OSLC: el lenguaje SRL se ofrece a través de una interfaz OSLC de entrada / salida, satisfaciendo la necesidad de reutilizar estándares en un entorno web.

Como se ha descrito anteriormente, y para combinar RDF y SRL, es necesario proporcionar una ontología RDFS / OWL, es decir, un vocabulario RDF, que defina las entidades y las relaciones en el modelo de representación de SRL para que esta especificación esté disponible públicamente y para permitir la expresión de cualquier conocimiento usando SRL. Por otro lado, y dado que una gran cantidad de datos, servicios y puntos finales basados en RDF y los principios de Linked Data ya están disponibles públicamente, un mapeo entre cualquier vocabulario RDF y SRL es completamente necesario para admitir la compatibilidad con versiones anteriores y poder importar cualquier pieza de datos RDF en RSHP. En este caso, considerando las pautas y definiciones de la especificación OSLC Core, la forma de los datos para la gestión del conocimiento se ajustará a las siguientes definiciones básicas de OSLC:

1. Un dominio OSLC es un área temática ALM o PLM. Cada dominio define una especificación.

2. Una especificación OSLC se compone de un conjunto fijo de recursos definidos por OSLC.

Los conceptos clave del metamodelo SRL son las clases Artefacto y Relaciones. Un artefacto es un contenedor de relaciones (RHSP) que puede tener metapropiedades

(autoría, versiones, características de visualización y, en general, información de procedencia) y expresiones de valor de atributo (AOV). Si un artefacto solo representa la aparición de un término, contendrá una referencia a un término (elemento de un vocabulario o taxonomía controlada). Este término puede tener una categoría gramatical (Tipo) como nombre, pronombre, adverbio o verbo para citar solo algunos. De la misma manera, una categoría semántica (Tipo) representada por un término puede asignarse a un término, por ejemplo, la semántica "negativa". Por lo tanto, diferentes términos pueden tener una semántica diferente. Finalmente, una relación establece un enlace entre n artefactos y la semántica también se puede adjuntar al enlace, p. "parte de".

3. Un recurso definido por OSLC es una entidad que se traduce en una clase RDF con un tipo. Cada recurso consta de un conjunto fijo de propiedades definidas cuyos valores pueden establecerse cuando el recurso se crea o actualiza.

En este caso, se ha definido una forma para cada clase. La siguiente tabla presenta los enlaces de forma de recurso a la definición oficial y una breve descripción del recurso.

Clase	**Descripción**
Artefact	Un contenedor de relaciones entre conceptos y metapropiedades para describir semánticamente cualquier pieza de información. Es la base para la creación de una red semántica subyacente.
Relationship	Una relación representa un enlace entre cualquier conjunto de recursos. Es posible agregar semántica y puede contener cualquier número de elementos que representen relaciones binarias, ternarias o incluso n-arias.
Data	Una expresión de valor de atributo que representa una propiedad del artefacto bajo descripción.
MetaData	Un atributo de valor de etiqueta que representa las propiedades típicas de metadatos. Pueden ser cualquier tipo de recurso o, más específicamente, conceptos.
Term	Este concepto sigue la semántica y la forma de un skos: Más específicamente: "la noción de un concepto SKOS es útil cuando se describe la estructura conceptual o intelectual de un sistema de organización del conocimiento, y cuando se hace referencia a ideas o significados específicos establecidos dentro de un KOS
Type	Todo tiene un tipo y un tipo es un tipo de concepto que proviene de una clasificación. P.ej. Los tipos de metamodelo UML, como Clase, Caso de uso, etc.

Tabla 11: Metadatos SRL

5.3.3. Reglas de transformación

Las reglas de transformación al usar VHDL son simples:

VHDL	**SRL**
Entity-Architecture pair	Artefact
Signals	Data (attibute)
Entity Ports	Data (attibute)
AND	Relationship
OR	Relationship
XOR	Relationship
NOT	Relationship
NAND	Relationship
NOR	Relationship
XNOR	Relationship
<	Relationship
>	Relationship
=	Relationship
<=	Relationship
>=	Relationship
/=	Relationship
+	Relationship
-	Relationship
*	Relationship
/	Relationship
**	Relationship
abs	Relationship
mod	Relationship
rem	Relationship
Nombre_clase	MetaData
BIT	Type
Boolean	Type
Std_logic	Type

Integer	Type
Bit_Vector	Type
Std_Logic_Vector	Type
Character	Type

Tabla 12: Transformación VHDL a SRL

Para hacer la transformación de VHDL a SRL hemos usado estas relaciones de forma genérica.

6. IMPLEMENTACION Y PRUEBAS

En este apartado se explicaran los distintos pasos que se han seguido para implementar todo el desarrollo del sistema descrito en el apartado 5.

6.1. Componentes del sistema

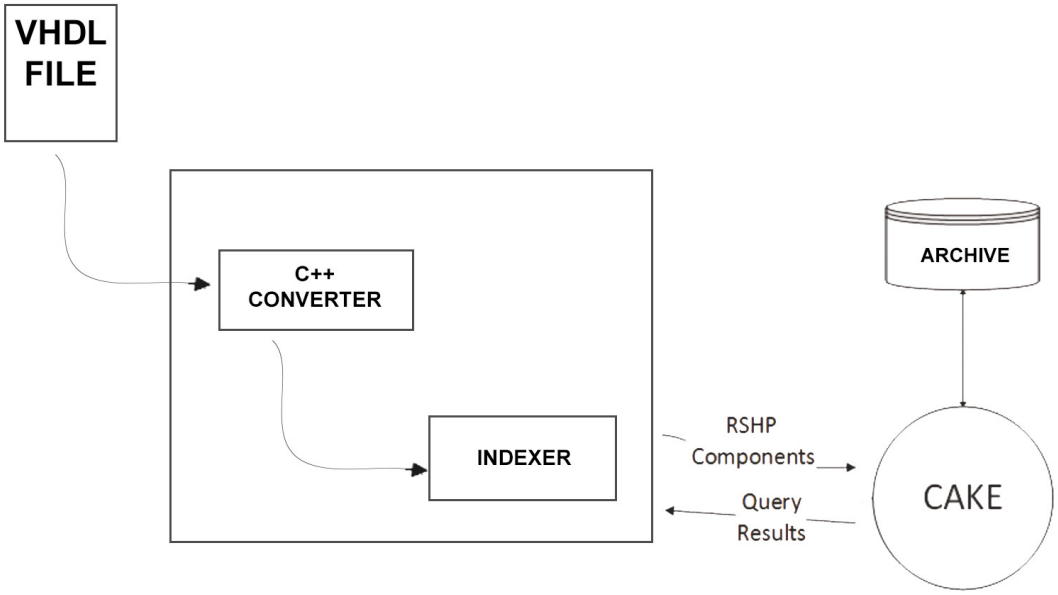


Ilustración 21: Componentes del sistema

Como se puede ver en la ilustración anterior, hay un componente encargado de transforma el lenguaje VHDL a SRL, otro componente recibe los datos de SRL y se los manda a Knowledge Manager que se encarga de gestionar la base de datos.

6.2. Diagrama de bloques y VHDL

En primer lugar, hemos analizado los datos sacados del diagrama de bloques de Quartus II en VHDL.

Para ello se ha hecho un programa en C++, en el cual se lee el archivo .vhd sacado por Quartus II e imprime por pantalla cada una de las entidades y arquitecturas del archivo para que más adelante sea posible la transformación.

En la siguiente imagen se puede ver un ejemplo de la impresión de leer un archivo en VHDL de un BCD.

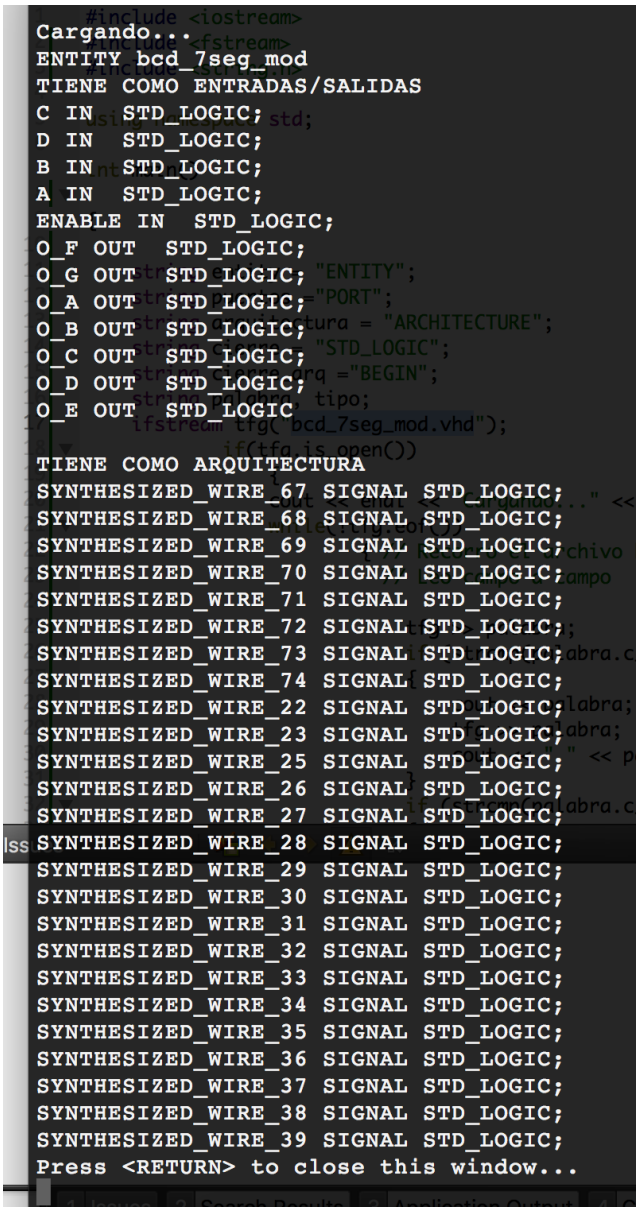


Ilustración 22: Captura de pantalla de la impresión del análisis de VHDL

Nuestra herramienta esta construida sobre .NET usando C++. Por eso se ha utilizado C++ para leer el archivo e imprimir por pantalla los datos leídos. Con esto pretendemos transformar lo procesado al leer el archivo VHDL en SRL (OSLC) para conseguir la reutilización deseada.

6.3.API CAKE

Para hacer la transformación de VHDL a SRL, se ha utilizado Cake como framework. La herramienta realiza los procesos de indexación y recuperación de información, que se pueden utilizar para analizar sus capacidades de reutilización.

Es el componente que admite el almacenamiento y la recuperación de información del modelo.

CAKE es un marco de herramientas, aplicaciones y metodologías para identificar, clasificar, organizar y reutilizar el conocimiento. El objetivo del marco es permitir que un usuario administre cualquier tipo de "activos de conocimiento". Utiliza RSHP como modelo de representación de información base.

Entre sus principales funcionalidades, CAKE admite la indexación y recuperación de información. La indexación de la información de un artefacto se realiza de acuerdo con los tipos de información y la estructura en RSHP. Cake también utiliza ontologías como bases de conocimiento de referencia que pueden derivar conocimientos adicionales. Esto es especialmente importante para la reutilización. Las ontologías de uso permiten a CAKE por ejemplo emplear sinónimos para la búsqueda de información. Los términos en un modelo RSHP son parte de las ontologías.

Para implementar las reglas de transformación, se tiene que llevar a cabo un proceso paso a paso. Teniendo en cuenta que RSHP y su tecnología subyacente (CAKE API1) se implementan en la plataforma .NET, hay que construir las fuentes. Posteriormente, se utilizó una herramienta de análisis para extraer las dependencias entre las diferentes bibliotecas de C++ y para generar un script que transformara las bibliotecas necesarias en DLL de .NET (biblioteca de enlace dinámico). Esto es suficiente enfoque fue suficiente para demostrar la posibilidad de integrar VHDL en la plataforma .NET. Por lo tanto, sería posible ofrecer un modelo de representación de información universal para indexar y recuperar metadatos y contenidos de modelos de sistemas físicos.

6.4. Knowledge manager

La herramienta KM funciona como repositorio de los datos.

“Un repositorio es un espacio centralizado donde se almacena, organiza, mantiene y difunde información digital, habitualmente archivos informáticos, que pueden contener trabajos científicos, conjuntos de datos o software”. [28]

Para proporcionar los servicios de gestión del conocimiento adecuados para los sistemas ciberfísicos, es necesario seleccionar un paradigma de representación del conocimiento adecuado. Obviamente, diferentes tipos de conocimiento requieren diferentes tipos de representación. En estas expresiones ligeras, los sistemas basados en reglas, las gramáticas regulares, las redes semánticas, las representaciones orientadas a objetos, , los agentes inteligentes o los modelos basados en casos, por nombrar solo algunos, son algunos de los principales enfoques para el modelado de información y conocimiento.

Más específicamente, la gestión del conocimiento también implica la estandarización de datos e información, es decir, cualquier bloque de información debe estructurarse y almacenarse para soportar otros servicios de aplicación.

La herramienta de gestión de conocimiento se puede utilizar para gestionar todos los artefactos generados.

6.5. Pruebas

En este apartado se quiere comprobar el correcto funcionamiento de la herramienta. Se van a realizar pruebas que permitan comprobar que se cumplen los requisitos que se definieron en el apartado X.

Para comprobar el correcto funcionamiento de la herramienta, se han hecho diferentes pruebas que se han documentado en tablas como la siguiente:

INDENTIFICADOR : PR – XX(1)	
REQUISITO RELACIONADO(2)	
DESCRIPCIÓN(3)	
RESULTADO ESPERADO(4)	
RESULTADO OBTENIDO(5)	

Tabla 13: Tabla ejemplo pruebas

(1) Identificador: Tiene un nombre único y permite identificar la prueba.

(2) Requisito relacionado: Identificador del requisito o requisitos que quiere comprobar la prueba.

(3) Descripción: Descripción de las acciones que se van a realizar para hacer la prueba.

(4) Resultado esperado: Se documenta el resultado esperado antes de hacer la prueba.

(5) Resultado obtenido: Contiene el resultado que se ha conseguido después de hacer la prueba.

INDENTIFICADOR : PR – 01	
REQUISITO RELACIONADO	R-01
DESCRIPCIÓN	Se comprueba que el sistema al hacer diagramas en Quartus II, lea el archivo .vhd
RESULTADO ESPERADO	Que Quartus II saqué un archivo de VHDL y que sea de texto (.vhd)
RESULTADO OBTENIDO	Cumple perfectamente el resultado esperado.

Tabla 14: Prueba PR-01

INDENTIFICADOR : PR – 02	
REQUISITO RELACIONADO	R-04
DESCRIPCIÓN	Se comprueba que el sistema sea capaz de leer
RESULTADO ESPERADO	Se debe imprimir por pantalla el mensaje de apertura de archivo
RESULTADO OBTENIDO	Cumple perfectamente el resultado esperado.

Tabla 15: Prueba PR-02

INDENTIFICADOR : PR – 03	
REQUISITO RELACIONADO	R-06
DESCRIPCIÓN	El sistema debe ser capaz de identificar e imprimir por pantalla los metadatos VHDL
RESULTADO ESPERADO	El sistema imprime la lista de entidades y arquitecturas con todos sus datos
RESULTADO OBTENIDO	Cumple perfectamente el resultado esperado.

Tabla 16: Prueba PR-03

INDENTIFICADOR : PR – 04	
REQUISITO RELACIONADO	R-07
DESCRIPCIÓN	El sistema debe transformar VHDL a SRL
RESULTADO ESPERADO	Mediante CAKE el sistema conseguirá reutilizar los datos
RESULTADO OBTENIDO	No se ha podido probar de forma práctica

Tabla 17: Prueba PR-04

7. MARCO LEGISLATIVO Y SOCIOECONÓMICO

En este punto se hará un breve análisis sobre el marco legislativo alrededor del proyecto.

También hablaremos del impacto socioeconómico que supone OSLC en el mundo y como mejorará la tecnología en un futuro.

7.1. Marco regulador y aspectos legales

En este proyecto nos afectan principalmente tres decretos: el Real Decreto 4/2010, en el que se habla de la interoperabilidad, ley de la propiedad intelectual y licencias de software. [22]

En primer lugar, el proyecto se basa en la interoperabilidad, para ello tenemos que tener en cuenta el Real Decreto 4/2010, que en su artículo 4 dice:

"La aplicación del Esquema Nacional de Interoperabilidad se desarrollará de acuerdo con los principios generales establecidos en el artículo 4 de la Ley 11/2007, de 22 de junio, y con los siguientes principios específicos de la interoperabilidad:

a) La interoperabilidad como cualidad integral.

b) Carácter multidimensional de la interoperabilidad.

c) Enfoque de soluciones multilaterales."

"La interoperabilidad se tendrá presente de forma integral desde la concepción de los servicios y sistemas y a lo largo de su ciclo de vida: planificación, diseño, adquisición, construcción, despliegue, explotación, publicación, conservación y acceso o interconexión con los mismos.

Se favorecerá la aproximación multilateral a la interoperabilidad de forma que se puedan obtener las ventajas derivadas del escalado, de la aplicación de las arquitecturas modulares y multiplataforma, de compartir, de reutilizar y de colaborar." [26]

Se ha hecho referencia a estos fragmentos, ya que nuestro proyecto básicamente busca lo que rige la ley. Buscamos la interoperabilidad siendo capaces de representar y reutilizar modelos hardware en VHDL. Teniendo presente desde el principio del proyecto el concepto de interoperabilidad y reutilización.

Por otro lado, la herramienta creada utiliza ficheros de otras organizaciones. Estos, están adheridos a la licencia de Creative Commons [23].

Debemos tener en cuenta que los datos se pueden adaptar y compartir, siempre que se reconozca la autoría y no se usen con finalidad comercial. Hay que tener en cuenta que todo lo que se haga con estos datos, va bajo la misma licencia.
También utilizamos una API creada por The Reuse Company [24], lo que nos obliga a cumplir los término y condiciones que se han implantado para su uso.

La medida legal que aplica el proyecto es la LPI (Ley de Protección Intelectual). Se atribuye a las personas involucradas en la creación de una obra el derecho exclusivo a la explotación de la obra, sin más limitaciones que las establecidas en la ley [25].

7.2. Entorno socioeconómico

Como ya hemos hablado antes, la cuarta revolución industrial, también llamada Industria 4.0, esta modificando la forma en la que operan los negocios, y por ello, se ven obligados a competir.

Esta revolución está ligada por la emersión de nuevas tecnologías, como por ejemplo la inteligencia artificial o la robótica. Las empresas pueden correr el riesgo de perder mercado si no se adaptan a los cambios de esta industria.

Para llevar a cabo negocios en un ciclo continuo desde diferentes localizaciones y fuentes, se necesita la integración digital de la información.

Para digitalizar cualquier empresa, las herramientas de integración juegan un papel muy importante, ya que ofrecen una mayor productividad y dotan a la empresa de una mayor productividad. Las empresas deben ser ágiles a la hora del desarrollo de aplicaciones inteligentes, estas metas se consiguen integrando los sistemas de forma óptima y teniendo un control de los datos completo.

Las herramientas de integración consiguen que las empresas tengan una mayor productividad y eficiencia, ya que conecta las diferentes herramientas de software, APIs y dispositivos para de esta forma poder automatizar los distintos procesos empresariales y ofrecer al usuario información exacta y precisa a tiempo.

La integración no solo conecta, sino que también añade valor a través de las nuevas funcionalidades provistas al conectar funciones diferentes del sistema. [16]

8. PLANIFICACIÓN

En este apartado se va a mostrar la planificación que se ha realizado desde que empezó el proyecto hasta que se acabó. El diagrama de Gantt está hecho por semanas, si se hacía por días era demasiado extenso y poco eficiente.

El proyecto comenzó la primera semana de abril y terminó aproximadamente para la entrega de la memoria, la tercera semana de septiembre.

Sin la planificación el desarrollo de un proyecto no es muy eficaz, ya que en con ella podemos identificar más fácilmente que tareas llevar a cabo y la secuencia de las mismas.

A partir del diagrama de Gantt se ha conocido las fechas aproximadas en las que se necesita tener cada parte del proyecto para poder completar satisfactoriamente cada una de las tareas.

En la siguiente ilustración se muestra el diagrama Gantt del proyecto:

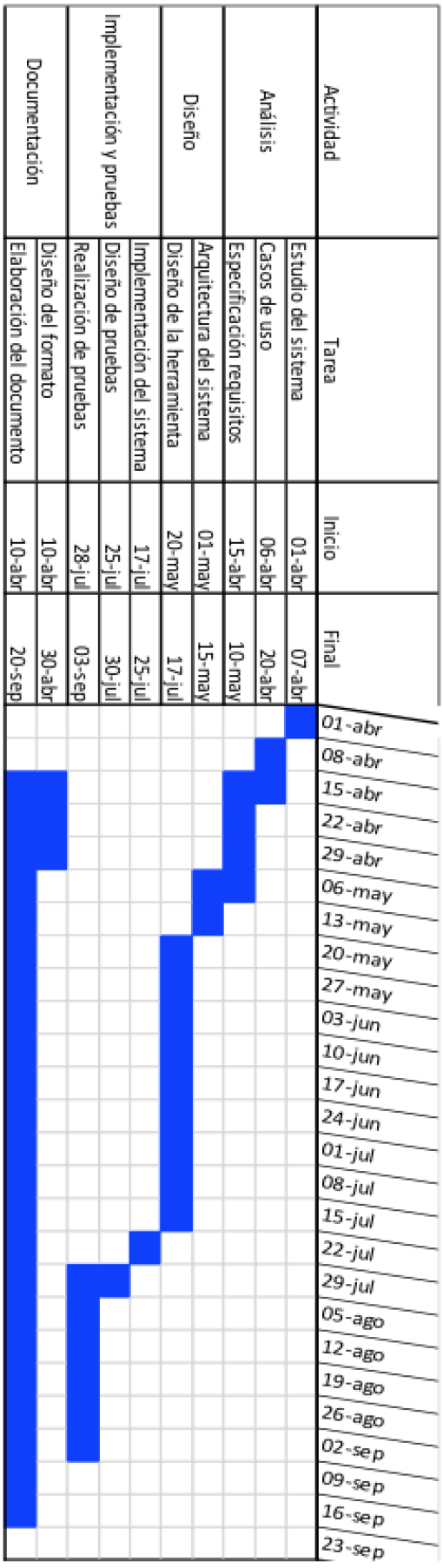


Ilustración 23: Diagrama de Gantt

En el diagrama de la ilustración anterior se puede ver que el desarrollo del proyecto comienza en abril y termina en septiembre. La documentación se va haciendo durante todo el desarrollo del proyecto, ya que es necesario ir documentando cada tarea. Existe una relación entre las tareas de Análisis, Diseño e Implementación y pruebas ya que como se puede observar para empezar la siguiente, se debe terminar la anterior.

9. PRESUPUESTO

En este apartado se va a realizar una estimación de los costes que ha supuesto el desarrollo de este proyecto. Se tendrá en cuenta el coste personal, de material empleado y costes indirectos.

9.1.Coste personal

Las personas implicadas en este proyecto, han sido:

Sara Martín Fraile: Desarrolladora de la herramienta y autora de la presente memoria.

José María Álvarez Rodríguez: Tutor del trabajo.

En este caso, no se ha tenido en cuenta el coste del tutor, aunque a parte de solventar dudas, fue el que propuso el tema y el que esta metido en la comunidad OSLC.

En las siguientes tablas se puede ver el coste del personal. Para la realización del coste se ha tenido en cuenta las horas trabajadas y se ha hecho una estimación de los costes según la tarea.

Se ha realizado una media de 2 horas por cada día trabajado en las distintas tareas. Para la memoria, en cambio, se ha trabajado una media de una 1 hora por día trabajado.

TAREA	**DÍAS**	**HORAS**
Análisis	25	50
Diseño	78	156
Implementación y pruebas	35	70
Documentación	115	115
TOTAL	148	391

Tabla 18: Total de días y horas trabajadas

Para la suma total de los días, no se ha tenido en cuenta la documentación, ya que por normal general la memoria se iba actualizando los días de trabajo en los que se elaboraban el resto de tareas.

Todas las tareas de la Tabla 18, no se realizan bajo un mismo rol. Por lo que cada uno tiene un coste por hora, como se ve en la Tabla 19, y a partir de eso se calcula el coste total del personal.

TAREA	ROL	COSTE €/HORA	TOTAL €
Análisis	Analista	25	1.250
Diseño	Programador	35	5.460
Implementación y pruebas	Tester	30	1.050
Documentación	Documentalista	20	2.300
TOTAL	-	-	10.060

Tabla 19: Coste personal

9.2. Coste de material

Para hacer un cálculo de los costes del material utilizado en el proyecto hay que hacer dos distinciones, costes hardware y costes software. Los primeros están relacionados con las máquinas utilizadas para desarrollar el proyecto y los segundos se relacionan con las herramientas empleadas para llevarlo a cabo.

9.2.1. Costes de Hardware

Las características del ordenador utilizado en el proyecto son:

- Fabricante: Apple
- Modelo: MacBook Pro (Retina 13 pulgadas, principios de 2014)
- Procesador: 2,2 GHz Intel Core i7
- Memoria: 16 GB 1600 MHz DDR3
- Gráficos: Intel Iris Pro 6100 1536 MB

Coste: 1.550 (mil quinientos cincuenta) euros.

Teniendo en cuenta de que la máquina ha sido utilizada durante 5 meses aproximadamente, calcularemos la amortización. Para ello suponemos que la amortización de un ordenador es de 48 meses.

$$Coste\ Hardware = 1449 * \frac{5}{48} = 161{,}46\ euros$$

9.2.2. Costes de Software

En este apartado tenemos que tener en cuenta que las licencias de las herramientas usadas han sido gratuitas ya que se trata de un trabajo académico, por lo que no se han producido costes de software en este sentido.

En cambio, si tenemos que tener en cuenta que se ha utilizado Microsoft Office 365 Personal, con un precio de 69 (sesenta y nueve) euros al año.

Para el cálculo de la amortización, hemos tenido en cuenta que la herramienta sólo ha sido usada unas 115 horas, aproximadamente una hora al mes durante 115 días (3,2 meses) , y que el periodo de amortización es de 36 meses:

$$Coste\ Software = \frac{69}{36} * 3{,}2 = 6{,}13\ euros$$

Resumiendo, en la siguiente tabla, se ven los costes del material

MATERIAL	**TIPO**	**PRECIO (€)**	**MESES**	**AMORTIZACION**	**TOTAL(€)**
Portátil	Hardware	1.550	5	48	161,46
Office	Software	69	3,2	36	6,13
TOTAL	-	-	-	-	167,59

Tabla 20: Coste material

9.3. Costes indirectos

Hemos tomado como costes indirectoS, la luz y la fibra y los desplazamientos:

COSTE	**PRECIO (€)/MES**	**MESES**	**TOTAL (€)**
Fibra y Luz	60	5	300
Desplazamiento	10	5	50
TOTAL	-	-	350€

Tabla 21: Costes indirectos

9.4. Coste total del proyecto

Finalmente, para el coste total del proyecto, se ha realizado la siguiente tabla, teniendo en cuenta todos los costes anteriores:

TIPO DE COSTE	**PRECIO (€)**
COSTES DE PERSONAL	10.060
COSTES DE MATERIAL	167,59
COSTES INDIRECTOS	350
TOTAL SIN BENEFICIOS	**10.577,59**
BENEFICIO ESPERADO (25%)	2.115,52
IMPORTE TOTAL CON BENEFICIO	**12.693,11**
IVA (21%)	2.665,55
TOTAL	15.358,66

Tabla 22: Coste total del proyecto

Como se puede observar, el coste total del proyecto es de 15.358,66 (quince mil trescientos cincuenta y ocho con sesenta y seis) euros.

10. CONCLUSIONES Y TRABAJO FUTURO

Para realizar este proyecto se ha tenido que llevar a cabo un gran trabajo de investigación, ya que no hay mucha información acerca de OSLC. Además, es un proyecto que no lleva muchos años y en el que todavía se pueden aportar muchísimas cosas.

El problema principal que hay ahora mismo es que no hay interoperabilidad entre herramientas, lo que supone un inconveniente a la hora de trabajar.

La comunidad OSLC puede marcar un antes y un después en la integración entre herramientas, es una forma ideal para poder trabajar sin ninguna traba y de forma mucho más eficaz. Se ha desarrollado para paliar viejos obstáculos y conseguir una integración efectiva de productos de ciclo de vida; OSLC permite crear integraciones a gran escala y de fácil mantenimiento en un entorno de herramientas heterogéneo.

Ya que muchos de los artefactos que se generan en procesos de ingeniería no gozan de un esquema definido. Circuitos eléctricos, modelos en 2D o modelos de simulación son algunos de los ejemplos que no tienen esquema definido.
Y que hay servicios que necesitan una visión global del sistema de desarrollo, y normalmente no están disponibles porque hay un bajo grado de conocimiento del sistema por la ausencia de intercambio de información. Queda muchísimo por investigar y desarrollar todo el tema de la interoperabilidad y la reutilización, solo estamos empezando un largo camino.

En este proyecto en concreto, se integrará todo el sistema de forma práctica para poder mostrar todos los detalles.

10.1. Motivación

La principal motivación por la que se ha realizado este trabajo es poder conseguir una herramienta que facilitaría mucho el trabajo en las empresas para poder compartir información.

Por otro lado, la motivación personal que me ha llevado a hacer este proyecto, ha sido querer desarrollar mis conocimientos sobre software, para poder aprender y profundizar más del tema. Algo que me parece muy importante para poder crecer profesionalmente.

En mi caso particular, trabajo en una consultora y me parece una forma buenísima de poder trabajar con el cliente de una forma mucho más rápida y eficiente, ya que no tendríamos que tener exactamente las mismas herramientas para trabajar y perderíamos menos tiempo en pasarnos información.

11. REFERENCIAS Y BIBLIOGRAFÍA

11.1. Referencias

[1]HTTPS://WWW.IBM.COM/SUPPORT/KNOWLEDGECENTER/ES/SSYMRC_6.0.2/COM.IBM.HELP.COMMON.OSLC.DOC/TOPICS/C_OSLC_OVERVIEW.HTML

[2]HTTPS://WWW.IBM.COM/SUPPORT/KNOWLEDGECENTER/ES/SSLKT6_7.6.0/COM.IBM.MT.DOC/GP_INTFRMWK/OSLC/C_OSLC_RES_SPEC.HTML

[3]https://es.wikipedia.org/wiki/Quartus_II

[4] https://www.cic.es/industria-40-revolucion-industrial/

[5] R. Baheti and H. Gill, "Cyber-physical systems," Impact Control Technol., vol. 12, pp. 161–166, 2011.

[6] N. Jazdi, "Cyber physical systems in the context of Industry 4.0," 2014, pp. 1–4.

[7] https://www.ticportal.es/expert/cps-cyber-physical-systems-implicaciones-desafios-futuros

[8]https://www.researchgate.net/publication/300466040_Reuse_of_Physical_System_Models_by_means_of_Semantic_Knowledge_Representation_A_Case_Study_applied_to_Modelica

[9] [https://es.wikipedia.org/wiki/Comunicación].

[10]http://www.para-agua.net/explorar/herramientas/sistemas-redes-informacion/p4-interoperabilidad

[11] K. Manikas and K. M. Hansen, "Software ecosystems – A systematic literature review," J. Syst. Softw., vol. 86, no. 5, pp. 1294–1306, May 2013.

[12] T. Thüm, S. Apel, C. Kästner, I. Schaefer, and G. Saake, "A Classification and Survey of Analysis Strategies for Software Product Lines," ACM Comput. Surv., vol. 47, no. 1, pp. 1–45, Jun. 2014.

[13] S. Powers, Practical RDF. Beijing ; Sebastopol: O'Reilly, 2003.

[14] V. Nguyen, O. Bodenreider, and A. Sheth, "Don't like RDF reification?: making statements about statements using singleton property," 2014, pp. 759–770.

[15] https://core.ac.uk/download/pdf/17036276.pdf

[16]https://revistabyte.es/tema-de-portada-byte-ti/software-integracion-la-herramienta/

[17] https://es.wikipedia.org/wiki/Microsoft_.NET

[18] https://es.wikipedia.org/wiki/Resource_Description_Framework

[19] https://www.reusecompany.com/km-knowledge-manager

[20] https://es.wikipedia.org/wiki/VHDL

[21] http://ayudaelectronica.com/que-es-vhdl/

[22]https://administracionelectronica.gob.es/pae_Home/pae_Estrategias/pae_Interoperabilidad_Inicio/pae_Esquema_Nacional_de_Interoperabilidad.html#.XYJqpy8RHUo

[23] https://creativecommons.org

[24] https://www.reusecompany.com/

[25] https: //www.boe.es/buscar/act.php?id=BOE-A-1996-8930

[26] https://www.boe.es/eli/es/rd/2010/01/08/4/con

[27] https://book.cakephp.org/1.3/es/The-Manual/Beginning-With-CakePHP/What-is-CakePHP-Why-Use-it.html

[28] https://es.wikipedia.org/wiki/Repositorio

[29] Documentos Universidad Carlos II de Madrid www.uc3m.es

11.2. Bibliografía

https://www.youtube.com/watch?v=-oXqudLmNMI

http://digitaldesignwithquartusii.blogspot.com/2013/10/el-entorno-de-desarrollo-de-altera-el.html

https://open-services.net/

mon.oslc.doc/topics/c_oslc_overview.html

mon.oslc.doc/topics/c_OSLC_create_int.html

https://open-services.net/resources/

https://www.ibm.com/support/knowledgecenter/es/SSHEB3_3.4.0/com.ibm.tap.doc_3.4.0/con_oslc/c_ctr_oslc_overview.html

https://www.ibm.com/support/knowledgecenter/es/SSLKT6_7.6.0.9/com.ibm.mif.doc/gp_intfrmwk/oslc/c_oslc_res_spec.html

https://es.slideshare.net/francisco.cifuentes/curso-ontologas-modelando-en-er-y-rdf-schema

https://www.seofreelance.es/que-es-rdf-introduccion-a-rdf/

https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=oslc-core

https://github.com/oasis-tcs/oslc-core

http://oslc.github.io/developing-oslc-applications/eclipse_lyo/eclipse-lyo.html

http://www.oasis-oslc.org/node/2

https://surf-vhdl.com/vhdl-syntax-web-course-surf-vhdl/vhdl-entity/

https://www.ibm.com/support/knowledgecenter/es/SSHEB3_3.4.0/com.ibm.tap.doc_3.4.0/con_oslc/c_ctr_oslc_provider_defn.html

.0/con_oslc/c_ctr_oslc_components.html

https://es.wikipedia.org/wiki/VHDL

https://es.wikibooks.org/wiki/Programación_en_VHDL/Introducción

https://e-archivo.uc3m.es/bitstream/handle/10016/24441/Towards_MODELSWARD_2017_69.pdf?sequence=1&isAllowed=y

https://www.researchgate.net/publication/300466040_Reuse_of_Physical_System_Models_by_means_of_Semantic_Knowledge_Representation_A_Case_Study_applied_to_Modelica

https://en.wikipedia.org/wiki/Open_Services_for_Lifecycle_Collaboration